IMAGE-AIDED NAVIGATION USING COOPERATIVE BINOCULAR STEREOPSIS

THESIS

Justin T. Soeder, Second Lieutenant, USAF

AFIT-ENG-14-M-70

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the United States Government.

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENG-14-M-70

IMAGE-AIDED NAVIGATION USING COOPERATIVE BINOCULAR STEREOPSIS

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Electrical Engineering

Justin T. Soeder, B.S.E.E.
Second Lieutenant, USAF

March 2014

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED

AFIT-ENG-14-M-70

IMAGE-AIDED NAVIGATION
USING COOPERATIVE BINOCULAR STEREOPSIS

Justin T. Soeder, B.S.E.E.
Second Lieutenant, USAF

Approved:

//signed// 6 Mar 2014
Dr. John F. Raquet, PhD (Chairman) Date

//signed// 3 Mar 2014
Dr. Kyle J. Kauffman, PhD (Member) Date

//signed// 3 Mar 2014
Dr. Gilbert L. Peterson, PhD (Member) Date

AFIT-ENG-14-M-70

Abstract

This thesis proposes a novel method for cooperatively estimating the positions of two vehicles in a global reference frame based on synchronized image and inertial information. The proposed technique – cooperative binocular stereopsis – leverages the ability of one vehicle to reliably localize itself relative to the other vehicle using image data which enables motion estimation from tracking the three dimensional positions of common features. Unlike popular simultaneous localization and mapping (SLAM) techniques, the method proposed in this work does not require that the positions of features be carried forward in memory. Instead, the optimal vehicle motion over a single time interval is estimated from the positions of common features using a modified bundle adjustment algorithm and is used as a measurement in a delayed state extended Kalman filter (EKF). The developed system achieves improved motion estimation as compared to previous work and is a potential alternative to map-based SLAM algorithms.

Table of Contents

Page

Abstract iv

Table of Contents v

List of Figures ix

List of Tables xi

List of Abbreviations xii

I. Introduction 1

1.1 Background 1
1.2 Problem 2
1.3 Scope 3
1.4 Related Work 4
1.5 Research Contributions 5
1.6 Research Outline 5
1.7 Thesis Outline 6

II. Background 7

2.1 Computer Vision 7
2.2 Vision-Aided Navigation 8
2.3 The Indoor Environment 9
2.4 Relevant Coordinate Frames 10
2.5 Feature Matching 13
  2.5.1 Feature Extraction 14
  2.5.2 Feature Description 14
  2.5.3 Feature Descriptor Matching 15
2.6 Model Fitting Using RANSAC 15
2.7 Kalman Filtering 17
  2.7.1 Linear Kalman Filter Propagation 17
  2.7.2 Delayed State Kalman Filter 19
2.8 SLAM 20
  2.8.1 Submapping Techniques 21
  2.8.2 Enforcing the Observability Conditions 23
  2.8.3 Inverse-Depth SLAM 23

- 2.9 Pose SLAM . . . 24
  - 2.9.1 Reduced Pose SLAM . . . 25
  - 2.9.2 Adaptive Windowing . . . 26
- 2.10 Image Geometry . . . 27
  - 2.10.1 Single View Geometry . . . 27
  - 2.10.2 Two Views . . . 28
  - 2.10.3 Three Views . . . 31
  - 2.10.4 n Views and Bundle Adjustment . . . 32
- 2.11 Stereo Vision SLAM . . . 33
- 2.12 Constraint Based Localization . . . 34
- 2.13 Hybrid Methods . . . 35
- 2.14 Extension to Multiple Vehicles . . . 36
- 2.15 Autonomous Navigation . . . 37
- 2.16 Previous Work . . . 38
  - 2.16.1 System Design and Dynamics . . . 38
  - 2.16.2 Current System . . . 40
- 2.17 Summary . . . 42

III. Methodology . . . 43

- 3.1 System Architecture . . . 43
  - 3.1.1 The ARDrone Platform . . . 44
  - 3.1.2 Hardware and Software . . . 45
    - 3.1.2.1 ROS Packages . . . 45
    - 3.1.2.2 Multiple Vehicle Network . . . 45
  - 3.1.3 Concept Diagram . . . 46
- 3.2 Target Tracking . . . 46
  - 3.2.1 Vehicle Targets . . . 46
  - 3.2.2 Tracking Loops . . . 48
- 3.3 Relative Camera Pose Estimation . . . 49
- 3.4 Feature Correspondence . . . 50
  - 3.4.1 Image Masking . . . 51
- 3.5 Feature Triangulation . . . 53
  - 3.5.1 Linear Triangulation . . . 54
  - 3.5.2 Measures of Accuracy . . . 55
- 3.6 Batch Processing . . . 55
  - 3.6.1 Feature Points . . . 57
  - 3.6.2 Relative Camera Position . . . 58
  - 3.6.3 Batch Solver . . . 59
    - 3.6.3.1 Levenburg-Marquardt Solution . . . 62
    - 3.6.3.2 Robust Cost Functions . . . 62
  - 3.6.4 Kalman Filter Measurement Formation . . . 63

Page

3.7 Navigation Kalman Filter . . . 64
  3.7.1 System Model . . . 64
  3.7.2 Vanishing Points . . . 67
  3.7.3 Delayed State Extended Kalman Filter Update . . . 67
3.8 Procedures . . . 69
  3.8.1 Camera Calibration . . . 69
  3.8.2 Tilt Calibration . . . 70
  3.8.3 Targeting Calibration . . . 71
  3.8.4 Data Collection . . . 72
3.9 Summary . . . 73

IV. Results . . . 74

4.1 Testing Equipment . . . 74
4.2 Feature Matching Experiment . . . 75
  4.2.1 Feature Matching Experiment Results . . . 75
4.3 Relative Camera Pose Estimation Accuracy Experiment . . . 77
  4.3.1 Relative Camera Pose Estimation Accuracy Experiment Results . . . 77
4.4 Triangulation Accuracy Experiment . . . 79
4.5 Batch Processing Experiment . . . 83
  4.5.1 Batch Processing Experiment Results . . . 84
4.6 Kalman Filter Experiments . . . 86
  4.6.1 Monte Carlo Simulations . . . 87
  4.6.2 Monte Carlo Simulation Results . . . 88
  4.6.3 Processor Integration Test Setup . . . 89
  4.6.4 Unfiltered Trajectory Estimate . . . 90
  4.6.5 Processor Integration Results . . . 92
  4.6.6 Re-Tuning the Kalman Filter . . . 97
  4.6.7 Localization Improvement . . . 100
4.7 Summary . . . 102

V. Conclusions . . . 103

5.1 Further Testing . . . 104
  5.1.1 Flight Path and Formation Control . . . 104
  5.1.2 Runtime Concerns . . . 104
  5.1.3 Feature Depletion . . . 105
  5.1.4 Comparison to State of the Art . . . 105
5.2 Future Work . . . 105
  5.2.1 Constrained Feature Matching . . . 105
  5.2.2 Absolute Position Measurements . . . 106
  5.2.3 Computational Improvements . . . 106

5.2.3.1 Sparse Linear Algebra 106
5.2.3.2 Parallelization 107
5.2.4 Improved Dynamic Model 107
5.3 Final Conclusion 108

References 109

List of Figures

Figure Page

2.1 Projective Geometry 8

2.2 Manipulating Indoor Structure for Determining Vanishing Points 10

2.3 Pixel Frame of the ARDrone's Forward Looking Camera 11

2.4 Camera Frame of the ARDrone 12

2.5 Body Frame of the ARDrone 13

2.6 Generic RANSAC Algorithm 16

2.7 An Illustration of Supmapping 22

2.8 Inverse Depth Parametrization 24

2.9 An Illustration of the Epipolar Constraint 28

2.10 Three Views of a Common Point 31

2.11 An Illustration of the Bundle Adjustment Problem 32

2.12 Separation of the Smoothing and Mapping Processes 36

2.13 System Design in the Previous Work 39

2.14 Results of the Previous Work 41

3.1 System Setup 44

3.2 Concept Diagram 47

3.3 Tracking Vehicle Targets 48

3.4 Results of Feature Correspondence 52

3.5 Image Processing Using Masks 52

3.6 Automatic Camera Calibration 70

3.7 Calibration of the Target Detector 72

4.1 Testing Equipment 74

4.2 Distributions of Relative Translation Errors 78

Figure Page

4.3 Distributions of Relative Orientation Errors 80

4.4 Sample Points for Evaluating Triangulation Accuracy 81

4.5 Feature Depth Error v. Parallax Angle 82

4.6 An Erroneous Match With Good Parallax Angle 82

4.7 Hallway Course Used for Testing the Batch Processing Algorithm 83

4.8 Batch Processor Solution Errors 85

4.9 Effect of Enforcing Solution Error Constraints 86

4.10 Normalized Measurement Error Distribution 87

4.11 Flight Profile for Monte Carlo Simulations 88

4.12 Monte Carlo Kalman Filter Simulation Results 89

4.13 Hallway Course Used for Testing the Integration of the batch processor and Kalman Filter 90

4.14 Trajectories Estimated From Measurements Only 91

4.15 Kalman Filter Position Estimate 92

4.16 Final Position Estimates 93

4.17 Kalman Filter Residuals v. Measurement Errors 94

4.18 Kalman Filter Residuals v. Measurement Errors (Continued) 95

4.19 Kalman Filter Velocity Estimates 96

4.20 Trajectory Estimates After Re-Tuning 98

4.21 Kalman Filter Velocity Estimates After Re-Tuning 99

4.22 Improvements Over Previous System 101

List of Tables

Table Page

4.1 Matching Parameters 76

4.2 Matching Performance 76

4.3 Final Position Errors Using Measurements Only 91

4.4 Final Position Error Using Kalman Filter 93

4.5 Final Position Errors After Re-Tuning 99

List of Abbreviations

Abbreviation		Page
GPS	Global Postioning System	1
IMU	Inertial Measurement Unit	1
SLAM	Simultaneous Localization and Mapping	4
DCM	Direction Cosine Matrix	12
SIFT	Scale Invariant Feature Transform	14
SURF	Speeded Up Robust Features	14
BRISK	Binary Robust Invariant Scalable Keypoints	15
EKF	Extended Kalman Filter	21
UKF	Unscented Kalman Filter	21
IEKF	Implicit Extended Kalman Filter	21
PnP	Perspective n-Point	27
SVS	Stereo Vision System	33
iSAM	incremental smoothing and mapping	36
GUI	Graphical User Interface	38
SDK	Software Development Kit	38
ROS	Robot Operating System	43
ROI	Region of Interest	48
GPU	Graphics Processing Unit	107

IMAGE-AIDED NAVIGATION
USING COOPERATIVE BINOCULAR STEREOPSIS

I. Introduction

Autonomous navigation in indoor and urban environments is a challenging problem. Arguably the most popular means of localization in many navigation systems involves the fusion of Global Postioning System (GPS) signals and the inertial signals of an intertial measurement unit (IMU). This technique capitalizes on an IMU's ability to capture quick motion and the ability of GPS to constrain long-term drift [20]. This approach, however, becomes degraded or unreliable in environments where GPS signals are either severely attenuated or denied – a typical characteristic of indoor and urban environments. The goal of this thesis is to develop a cooperative indoor navigation algorithm for a system of two vehicles which are able to communicate image and navigational data with each other.

1.1 Background

With the recent expansions in computational capability and the streamlining of computer vision algorithms, visual sensors (e.g. cameras) have become an attractive substitute for GPS receivers in the sensor-aided IMU framework. Visual sensors provide a number of benefits, such as low cost and weight. These sensors are also able to measure distinct features in the environment, which is useful for constructing a map of the environment.

The ARDrone quadrotor helicopter, developed by Parrot SA, is a commercially available platform which sports a number of navigational sensors, including a forward looking camera, an optical flow sensor, altitude sensors, and inertial sensors [1]. The ARDrone also comes equipped with an on-board processor which enables wireless communication, filters inertial sensor measurements, and stabilizes the vehicle while airborne. The abundant sensors and processor features, as well as ease of use, make the ARDrone a viable tool for developing image-aided navigation algorithms.

1.2 Problem

Previously, Dean [18] developed a real-time image processing algorithm to extract heading cues from the image stream of the ARDrone's forward looking camera. These heading cues were used to constrain rotational errors in the vehicle's filtered navigation estimates and correct directional errors in the velocities computed by the vehicle's on-board image processing algorithm. However, while the heading error was constrained, it was found that the optical flow algorithm running on-board returned a poor estimate of the vehicle's velocity, which ultimately led to a poor estimate of the vehicle's trajectory. In addition, the optical flow algorithm required that texture be added to the flight path in order to return a reasonable estimate, which is not feasible for an autonomous system.

This research is focused on developing a new image processing algorithm which more accurately estimates the ARDrone's trajectory without having to alter the operating environment. The developed algorithm leverages additional sensor information from a supporting ARDrone operating within its field of view and processes the image and onboard navigation information from both vehicles to localize both vehicles in a global reference frame.

1.3 Scope

The central idea in this thesis is that camera motion can be estimated from tracking the collective motion of features through a sequence of images. A feature is a point in the environment that can be distinguished from the area surrounding it – e.g. corners and edges. Feature tracking consists of determining corresponding image projections of the same feature in separate images. Assuming that each tracked feature is stationary in the global reference frame, the change in its projection onto the image plane must be a result of camera motion.

Feature localization in this research is achieved through a cooperative adaptation of binocular stereopsis. In traditional binocular stereopsis, a stereo pair of cameras is mounted on a single vehicle and calibrated prior to flight to determine the relative orientation and baseline between cameras, as well as the individual camera calibration matrices [28]. These parameters can be used to determine the three dimensional coordinates of feature points common to both images. In this research, two vehicles are flown in formation such that a leading vehicle can be identified in the corresponding image of the trailing vehicle. The relative translation and orientation for the pair of cameras can be re-estimated for each set of corresponding images – essentially recalibrating the stereo pair for each set of images. Features can then be tracked over consecutive image pairs to determine the motion of each vehicle.

The system developed in this thesis will be confined to operate indoors. Specifically, the algorithms developed in this thesis and taken from previous work [18] take advantage of the structural properties of hallways to achieve localization. In addition, the lighting conditions and amount of detectable features will be assumed not to be an issue for algorithm developed which performs feature tracking.

1.4 Related Work

Motion estimation by feature tracking is typically implemented in a technique called simultaneous localization and mapping (SLAM). A SLAM algorithm includes the coordinates of identified features as random variables in an estimator. Over time, measurements will reduce the uncertainty in each tracked feature's location. Theoretically, as the uncertainty in a feature's location is minimized, the feature will become a landmark and the vehicle's position uncertainty will decrease. SLAM was first developed in [59] and has been explored in works such as [68], [14], [19], and [56].

While theoretically sound, SLAM algorithms in practice are prone to inconsistency [36] [5]. Inconsistency is defined as the condition in which the error covariance of the vehicle's position estimate is underestimated. In addition, the large structure of the SLAM problem is not typically feasible for small vehicles with limited payload for computing power. Submapping techniques are a subset of the larger SLAM literature which seek to reduce computation and error by dividing an environment into maps populated by feature points [50] [51] [13] [22]. In Pose SLAM [32] [38] [41], the features are eliminated altogether and the map consists of a finite or infinite number of past vehicle positions.

Alternatively to maintaining all tracked features in a SLAM network, the motion of a feature over time can be used in a navigation estimator without including its position as a random variable. In [48], [7], and [20] the changing locations of features across images were used to derive constraints on vehicle motion. In [48] specifically, the difference in the location of a feature is related to the current vehicle position and a history of previous vehicle positions. Each feature produces a constraint on the positions of the vehicle over time, similar to the algorithm developed in this thesis. In contrast, the algorithm developed in this thesis handles all tracked features simultaneously to produce a single measurement which corresponds to the difference between each vehicle's current position and its position at the previous time-step.

Another key feature of this work is the cooperative aspect. Cooperative localization has been explored in a number of different facets. In [61] and [19], a network of vehicles navigates in an alternating fashion where vehicles take turns acting as landmarks for the rest of the vehicles in the group to navigate. This approach is generally not feasible for airborne vehicles. In [45] and [34], all of the vehicles move continuously and the relative position of each vehicle is inferred from features that are common to both of their cameras. The system designed in this thesis uses a combination of these two approaches.

1.5 Research Contributions

This research expands the developed system model outlined in [18] to enable estimation of the three dimensional position of two vehicles. This research also contributes a cooperative localization algorithm and a flexible state estimation algorithm, based on the expanded system model, which is able to handle a varying number of measurements.

1.6 Research Outline

The research begins by modifying existing software to support a second vehicle and multiple vehicle communications. With the expanded software structure in place, the cooperative binocular stereopsis algorithm is developed to capture and process a stereo image pair and return a structure of image measurements. These measurements include vanishing points, as described in [18]; the estimated camera baseline; and a varying number of three-dimensional feature points, found through linear triangulation. As an intermediate step, the estimated baseline and feature points are processed simultaneously to estimate the change in position and attitude for each vehicle between two time epochs using nonlinear regression. The state estimation algorithm is then developed to handle the measurements in a delayed state Kalman filter. A series of experiments are then presented to validate the system in stages: starting from validating the image processing algorithm, including feature matching, and leading up to validating the full Kalman filtering algorithm.

1.7 Thesis Outline

Chapter 2 provides background information on the topics of vision aided navigation, computer vision, autonomous navigation, and cooperative vehicle behavior. The purpose is two-fold: to introduce mathematical concepts and conventions that are fundamental to this thesis and to provide an analysis of current and past approaches to the localization problem. Chapter 3 develops the algorithms, code architectures, and system model used in this research. Chapter 4 describes the experiments used to test the developed system and presents their results. Chapter 5 contains conclusions and suggestions for further research.

II. Background

This chapter discusses relevant topics in vision navigation as they apply to the research described in this thesis. The chapter begins by introducing computer vision as it applies to navigation using vision. Next, the area of navigation using vision is described – focusing on the case of vision-aiding for indoor navigation. Techniques for feature matching and outlier rejection are then introduced. Next, the Kalman filter equations used in this thesis are presented. Various localization mechanisms are then presented, including SLAM, multi-view geometric, and batch methods. The topic of localization is then extended to the case of multiple cooperative vehicles and a brief discussion of autonomous systems is presented. This thesis follows the work presented in [18], in which perspective features were used to correct heading errors for a single navigating quadrotor. The chapter closes with an overview of the system designed in [18].

2.1 Computer Vision

A core component of this research is computer vision. Fundamentally, computer vision is derived from projective geometry, where 3D space is captured on an infinite 2D plane. In computer vision, however, the dimensions and projection resolution of the 2D plane are limited by the visual sensor. An illustration of projection as it applies to computer vision is shown in Figure 2.1.

Computer vision seeks mathematical analogues to perception. For instance, sharp edges in an image correspond to high spatial frequencies. Colors in an image can be represented by a vector describing the combination of a basis color set – such as red, green, and blue (RGB). These mathematical interpretations are what enable artificial perception in digital systems.

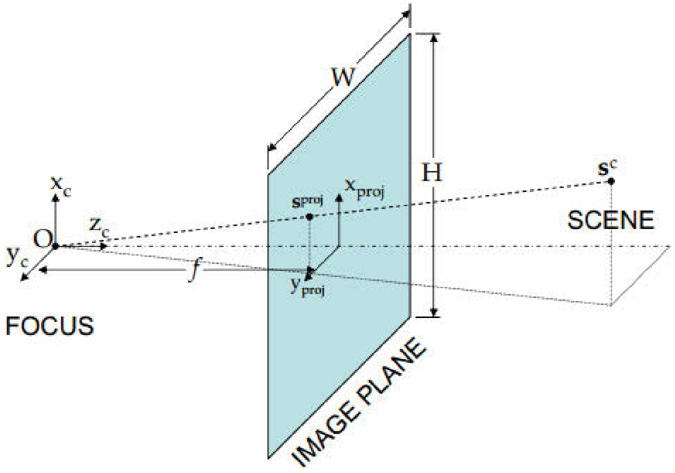


Figure 2.1: Projective Geometry. 3-dimensional points from a scene are represented on a 2-dimensional plane. The image plane in computer vision is constrained by physical dimensions *W* and *H* [66].

2.2 Vision-Aided Navigation

It is important to make a subtle but important note on the similarities and differences between vision-based and vision-aided navigation systems. Vision-based navigation systems seek to localize a body based only on visual data. Techniques that fall into the vision-based category include structure from motion and bundle adjustment [21] [33] [65]. An example of a vision-based system is [54], where scene analysis is used to extract significant landmarks and enable goal-seeking navigation. Vision-based navigation systems can be limited by the computational cost of each image processing algorithm, the accuracy of the imaging sensors, and the accuracy of the updating mechanism.

In contrast, a vision-aided navigation system synthesizes visual data to aid a dynamic model. A dynamic model provides an estimate of position based on the equations of motion. A simple dynamic model takes the form

$$\dot{\begin{bmatrix} X \\ Y \\ V_x \\ V_y \end{bmatrix}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ V_x \\ V_y \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta v_a \\ \delta v_b \end{bmatrix} \tag{2.1}$$

where X,Y are positions, V_x, V_y are velocities, and δv_a,δv_b are velocity correction inputs to the system. Visual data is then used to refine the model's estimate. Examples of vision aided navigation algorithms are found in [35], [34], and [42].

2.3 The Indoor Environment

It is also important to characterize the environment in which a vision-aided navigation system will operate. In an indoor environment, the geometry of man-made structures can be exploited for navigation purposes. In the previous work [18], the known orthogonality of the indoor environment was exploited to calculate the theoretical intersection of parallel lines – the vanishing point. An illustration of this idea is shown in Figure 2.2.

The disadvantage of the indoor environment, from a feature tracking perspective, is that a blank section of a wall might only provide sparse features which are difficult to identify or match over time. Additionally, dimly lit structures can cause problems for imaging sensors. These problems will be assumed negligible in this research by operating in well-lit environments.

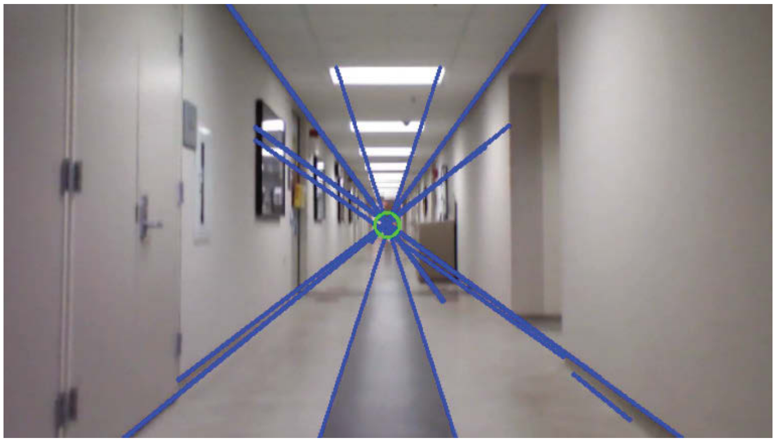


Figure 2.2: Manipulating Indoor Structure for Determining Vanishing Points. The vanishing point algorithm in [18] hinges on the assumption that the intersecting lines shown are projections of parallel lines in an indoor landscape.

2.4 Relevant Coordinate Frames

The following coordinate frames are defined for the reader as they are used throughout this document. These frames were defined in [66], and are presented here merely for completeness.

Pixel Frame: The pixel frame$\{x^{pix}, y^{pix}\}$ is the image plane shown in Figure 2.1. It is a two-dimensional frame in which the projected coordinates of a 3D point are captured. The x and y axes represent the columns and rows, respectively, expressed in pixels. In this document, the origin of the pixel frame is defined as the upper left-hand corner of the pixel frame, as shown in Figure 2.3.

Camera Frame: The camera frame $\{X^c, Y^c, Z^c\}$ is a three-dimensional, orthogonal, right-handed frame with its origin at the center of the camera lens. In this frame, the positive z-axis is directed outward from the camera lens, the positive y-axis is directed downward, and the positive x-axis is directed to the right. An illustration is shown in Figure 2.4. The camera frame is related to the pixel frame by the equation

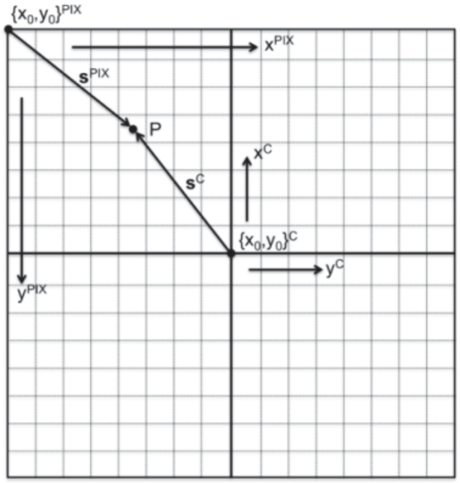


Figure 2.3: Pixel Frame of the ARDrone's Forward Looking Camera. Three-dimensional points from a scene are projected onto a two-dimensional plane and are given coordinates in pixels [18].

$$\begin{aligned}\mathbf{s}^{pix} &= \frac{1}{[\mathbf{s}^c]_z}\mathbf{T}_c^{pix}\mathbf{s}^c \\ \mathbf{T}_c^{pix} &= \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}\end{aligned} \tag{2.2}$$

where $\mathbf{s}^c$ is a vector to a target in the camera frame and $\mathbf{T}_c^{pix}$ is the camera calibration matrix which has the following parameters: the x and y focal lengths f_x, f_y; the skew γ; and the camera center coordinates c_x, c_Y in pixels. These parameters are determined through camera calibration. Note that $[\mathbf{s}^c]_z$ denotes the z component of $\mathbf{s}^c$.

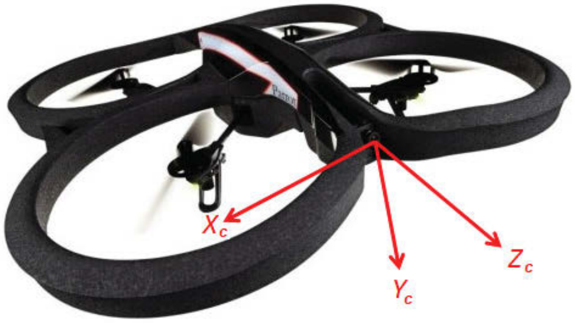


Figure 2.4: Camera Frame of the ARDrone. The camera frame is a right handed frame with the origin located at the center of the ARDrone's foward looking camera.

Body Frame: The body frame $\{X^b, Y^b, Z^b\}$ is attached to the vehicle and is the frame in which measurements are resolved. The Euler angles roll, pitch, and yaw (ϕ, θ, ψ) are defined as a sequence of rotations about the x, y, and z-axes, respectively, of the body frame. The body frame is superimposed on the ARDrone airframe in Figure 2.5. The body frame is related to the camera frame by the coordinate transform

$$\mathbf{C}_c^b = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.3}$$

where $\mathbf{C}_c^b$ is a direction cosine matrix (DCM) [63] which rotates points in the camera frame into the body frame.

Navigation Frame: The navigation frame $\{X^n, Y^n, Z^n\}$ is an earth fixed, local level frame. The origins of the body and navigation frame coincide when the drone is initialized, but the navigation frame remains fixed as the body frame rotates and translates in flight. Quantities will ultimately be expressed in the navigation frame, as this frame

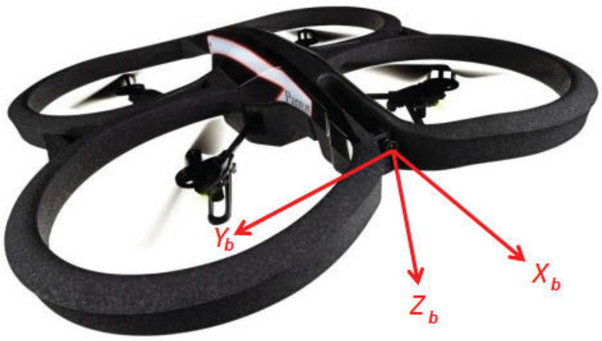


Figure 2.5: Body Frame of the ARDrone. In this frame, the *x*-axis extends out from the nose of the vehicle, the *y*-axis extends out the right wing, and the *z*-axis extends downward [18].

provides meaningful information to the user. The navigation frame is related to the body frame by the coordinate transform [63]

$$\mathbf{C}_b^n = \begin{bmatrix} \cos(\theta)\cos(\psi) & -\cos(\phi)\sin(\psi)+ & \sin(\phi)\sin(\psi)+ \\ & \sin(\phi)\sin(\theta)\cos(\psi) & \cos(\phi)\sin(\theta)\cos(\psi) \\ \cos(\theta)\sin(\psi) & \cos(\theta)\cos(\psi)+ & -\sin(\phi)\cos(\psi)+ \\ & \sin(\phi)\sin(\theta)\sin(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix} \tag{2.4}$$

where $\mathbf{C}_b^n$ is also a DCM.

2.5 Feature Matching

Feature matching is the process by which points in separate images are determined to belong to the same three dimensional feature. This process consists of three stages: extraction, description, and descriptor matching. There are a number of algorithms that exist for all three stages and, for the most part, these algorithms can be combined in any number of ways.

2.5.1 Feature Extraction.

Feature extraction is the process by which distinguishable points or shapes in an image are identified according to a specific criterion. The goal of feature extraction is to sample an image for keypoints which have a high probability of being found in another image. Feature extraction algorithms can be classified into two main categories: local intensity methods and spatial frequency methods.

Local intensity methods – such as the Harris corner detector, Lucas-Kanade tracker, and others [58][25] – identify regions in an image where pixel intensities change rapidly by calculating the eigenvalues of a matrix of image gradients. As image detail increases, so do the eigenvalues of this matrix and therefore keypoints can be identified by thresholding the eigenvalues [58].

Identifying keypoints which have a large change in intensity does not guarantee that the keypoints will be reliably identified in another image; a more robust method is desired. As an example, consider the scale invariant feature transform (SIFT) algorithm developed by Lowe [43]. SIFT uses a difference-of-Gaussian to find stable keypoints. An advantage of this algorithm is its ability to find keypoints which will be invariant to scale changes, which makes this algorithm very popular; however, the algorithm will still fail under large affine transforms. A typical image can return nearly 2000 stable SIFT keypoints [43].

2.5.2 Feature Description.

The goal of feature description is to describe a feature point using the image area surrounding it. This description should be unique enough to distinguish it from other features, but generic enough to allow the same feature to be described in the same way under changing environment conditions (e.g. scale, rotation, and lighting). For example, SIFT has a 128-element Euclidean descriptor vector determined by the local image gradient surrounding the feature point [43]. The SURF algorithm, a SIFT variant, detects features based on the Hessian matrix and integral images, resulting in feature descriptors which are more robust than SIFT and only have 64 elements [6].

Recently, binary descriptors have become a more attractive option for feature description in real-time applications. As its name would suggest, a binary descriptor seeks to uniquely define a point in an image using a vector of binary values. Binary descriptors are attractive from a computational perspective, since computing the distance between two binary vectors simplifies to boolean arithmetic. An example of a binary descriptor algorithm is the binary robust invariant scalable keypoints (BRISK) algorithm [40]. The BRISK descriptor is constructed by thresholding the intensity of points sampled near the feature of interest. Points which exceed the threshold are given a value of 1, and points below the threshold are given value 0.

2.5.3 Feature Descriptor Matching.

Feature descriptor matching is the final process in feature matching which attempts to find a set of matching keypoints based on the computed distance between descriptors. For Euclidean descriptors (e.g. SIFT) the distance is the vector norm of the difference of the two descriptor vectors. For binary descriptors (e.g. BRISK) the distance is computed by an XOR operation of each corresponding element of the two descriptor vectors followed by a bit count [40]. The latter is much faster to compute than a vector norm.

2.6 Model Fitting Using RANSAC

Consider the problem of determining the outliers in a series of (X, Y) data points. A naive approach would be to determine the line of best fit using all data and then eliminate the outliers. Including all data can greatly skew the line of best fit. In the RANSAC approach [24], the line of best fit would be estimated from a small, random subset of the entire dataset, which reduces the impact of outliers.

At each iteration, the RANSAC algorithm determines a model for the entire set based on a random subset of data. The number of data points that agree with the model – the inliers – are counted. If there are enough inliers, the model is accepted and the algorithm is finished. Otherwise, the algorithm iterates through another subset or ends in failure if the maximum number of iterations is reached. A generic RANSAC algorithm is presented in Figure 2.6.

```
procedure Ransac
Require: a mapping $M(B \leftarrow A)$
    given data sets $A$ and $B$
    parameters $n$ = subset size, $\tau$ = threshold, $i$ = max iterations
    $N$ = set size for consensus
    select random set $C \subset A$ of size $n$
    repeat
        determine a mapping, $M(B \leftarrow C)$
        $\forall\ \alpha \in A, \beta \in B$
        $A^* = \{\alpha \ni ||M(\alpha) - \beta|| > \tau\}$
        if size($A^*$) > $N$ then
            $M(B \leftarrow A) = M()$
            done
        else if number of iterations < $i$ then
            select new random $C \subset A$ of size $n$
        else
            end in failure
            done
        end if
    until done
end procedure
```

Figure 2.6: Generic RANSAC Algorithm. Rather than use all data, RANSAC attempts to find a model fit from a random subset of data points. This method is more resistant to outliers than standard least squares estimation.

In the context of the feature matching scenario in the previous section, RANSAC proves to be very useful for pruning out erroneous feature matches. Feature extraction and matching produces an initial pairing of points in each view, of which a handful of matches may be incorrect. After defining a mathematical mapping between the pairs of points (such as the fundamental matrix), RANSAC can be used to simultaneously determine the best mapping and remove feature correspondences which do not fit this mapping.

2.7 Kalman Filtering

A Kalman filter is a recursive algorithm which provides an estimate of the mean and covariance of a stochastic differential equation based on the previous estimate, deterministic inputs, measurements, and independent white Gaussian noises. There are a number of variations of the Kalman filter derived in literature. For this thesis, the Kalman filter will be described as a two step process consisting of a linear propagation [44] and a delayed state update [11].

2.7.1 Linear Kalman Filter Propagation.

A Kalman filter is derived from a stochastic differential equation of the form

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) + \mathbf{G}(t)\mathbf{w}(t) \tag{2.5}$$

where $\mathbf{F}(t)$ is the dynamics matrix, $\mathbf{x}(t)$ is the state of the equation, $\mathbf{B}(t)$ is the input matrix, $\mathbf{u}(t)$ is a vector of inputs, $\mathbf{G}(t)$ is the noise matrix, and $\mathbf{w}(t)$ is a vector of white Gaussian noise sources having the properties

$$\begin{aligned} E[\mathbf{w}(t)] &= \mathbf{0} \\ E[\mathbf{w}(t)\mathbf{w}(t')^T] &= \mathbf{Q}(t)\delta(t-t') \end{aligned} \tag{2.6}$$

where $E[\cdot]$ is the expectation operator, $\delta(t - t')$ is the dirac delta function, and $\mathbf{Q}(t)$ is a matrix of noise strengths at time t. In the discrete-time implementation [44], propagation is defined by the equation

$$\hat{\mathbf{x}}(t_k^-) = \mathbf{\Phi}(t_k, t_{k-1})\hat{\mathbf{x}}(t_{k-1}^+) + \mathbf{B}_d(t_k)\mathbf{u}(t_{k-1}) \tag{2.7}$$

where the $\hat{\mathbf{x}}(t_k^-)$ is the propagated state estimate at time t_k prior to a measurement update (indicated by the superscript minus) and $\hat{\mathbf{x}}(t_{k-1}^+)$ is the updated state at time t_{k-1} (indicated by the superscript plus). The state transition matrix $\mathbf{\Phi}(t_k, t_{k-1})$ is defined as [44]

$$\mathbf{\Phi}(t_k, t_{k-1}) = e^{\mathbf{F}(t_k)\cdot(t_k - t_{k-1})} \tag{2.8}$$

and can be approximated by

$$\mathbf{\Phi}(t_k, t_{k-1}) = \mathbf{I} + \mathbf{F}(t_k) \cdot (t_k - t_{k-1}) \tag{2.9}$$

when the time difference between t_k and t_{k-1} is relatively small compared to the dynamics equations. Similarly, the approximation [44]

$$\mathbf{B}_d(t_k) = \mathbf{B}(t_k) \cdot (t_k - t_{k-1}) \tag{2.10}$$

for the discrete input matrix $\mathbf{B}_d(t_k)$ can also be made when the time difference is small.

The covariance of the state vector after propagation $\mathbf{P}(t_k^-)$ is computed by the equation [44]

$$\mathbf{P}(t_k^-) = \mathbf{\Phi}(t_k, t_{k-1})\mathbf{P}(t_{k-1}^+)\mathbf{\Phi}^T(t_k, t_{k-1}) + \mathbf{Q}_d(t_k) \tag{2.11}$$

where the approximation

$$\mathbf{Q}_d(t_k) = \mathbf{G}(t_k)\mathbf{Q}(t_k)\mathbf{G}^T(t_k) \cdot (t_k - t_{k-1}) \tag{2.12}$$

for the discrete noise matrix $\mathbf{Q}_d(t_k)$ can be made when the time difference is small.

2.7.2 Delayed State Kalman Filter .

Traditionally, a Kalman filter represents measurements as a function of the propagated state $\hat{\mathbf{x}}(t_k^-)$. However, in some situations, it is more appropriate to relate the measurement to the propagated state at time k and the updated state at time $k-1$. This measurement equation takes the form [11]

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}(t_k^-), \mathbf{x}(t_{k-1}^+)) + \mathbf{v}_k \tag{2.13}$$

where $\mathbf{z}_k$ is the realized measurement, $\mathbf{h}(\mathbf{x}(t_k^-), \mathbf{x}(t_{k-1}^+))$ is a nonlinear measurement function, and $\mathbf{v}_k$ is additive white Gaussian noise with noise strength $\mathbf{R}_{zz}$. The update equation is [11]

$$\hat{\mathbf{x}}(t_k^+) = \hat{\mathbf{x}}(t_k^-) + \mathbf{K}[\mathbf{z}_k - \mathbf{h}(\mathbf{x}(t_k^-), \mathbf{x}(t_{k-1}^+))] \tag{2.14}$$

where $\mathbf{K}$ is the Kalman gain matrix computed by the equations [11]

$$\begin{aligned} \mathbf{L} =& \mathbf{H}_k\mathbf{P}(t_k^-)\mathbf{H}_k^T + \mathbf{R}_{zz} + \mathbf{H}_{k-1}\mathbf{P}(t_{k-1}^+)\mathbf{H}_{k-1}^T \\ &+ \mathbf{H}_k\mathbf{\Phi}\mathbf{P}(t_{k-1}^+)\mathbf{H}_{k-1}^T + \mathbf{H}_{k-1}\mathbf{P}(t_{k-1}^+)\mathbf{\Phi}^T\mathbf{H}_k^T \qquad (2.15) \\ \mathbf{K} =& \mathbf{P}(t_k^-)\mathbf{H}_k^T + \mathbf{\Phi}\mathbf{P}(t_{k-1}^+)\mathbf{H}_{k-1}^T\mathbf{L}^{-1} \qquad (2.16) \end{aligned}$$

and the matrices $\mathbf{H}_k$ and $\mathbf{H}_{k-1}$ are the first order linearizations of the function $\mathbf{h}(\mathbf{x}(t_k^-), \mathbf{x}(t_{k-1}^+))$ with respect to $\mathbf{x}(t_k^-)$ and $\mathbf{x}(t_{k-1}^+)$. The covariance update is computed from the equation

$$\mathbf{P}(t_k^+) = \mathbf{P}(t_k^-) - \mathbf{K}\mathbf{L}\mathbf{K}^T \tag{2.17}$$

Equations 2.15, 2.16, 2.17 form the basis for the delayed state Kalman filter.

2.8 SLAM

Fundamentally, a SLAM algorithm attempts to simultaneously solve for the position of a vehicle and create a map of landmark features in an unknown environment. The seminal work in this area was proposed by Smith, et al. [59], who showed that spatial information could be represented in a stochastic map. Consider the simplified example in which the position and orientation of a robot is described by two dimensional position coordinates (x, y) and an angle ϕ defined as a rotation about an orthogonal z-axis such that the vehicle state is

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \phi \end{bmatrix} \tag{2.18}$$

In this example, uncertain spatial relationships are modelled by the mean and covariance of the state $\mathbf{x}$

$$\hat{\mathbf{x}} = E(\mathbf{x}) = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\phi} \end{bmatrix} \quad \mathbf{P} = E\left[(\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})^T\right] = \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 & \sigma_{x\phi}^2 \\ \sigma_{xy}^2 & \sigma_y^2 & \sigma_{y\phi}^2 \\ \sigma_{x\phi}^2 & \sigma_{y\phi}^2 & \sigma_\phi^2 \end{bmatrix} \tag{2.19}$$

The distribution which assumes the least information, given only the mean and covariance, is the normal distribution [59]. Thus, the SLAM algorithm can be developed as a Kalman Filter. As a vehicle navigates through an environment, new features are detected and augmented to the state if they pass some criterion. Features are usually given a position in the same coordinate frame as the filter state such that at any given time the state and covariance of the filter are

$$\hat{\mathbf{x}} = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{x}_{f1} \\ \mathbf{x}_{f2} \\ \vdots \\ \mathbf{x}_{fn} \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} \mathbf{P}_R & \mathbf{P}_{R,f1} & \mathbf{P}_{R,f2} & \cdots & \mathbf{P}_{R,fn} \\ \mathbf{P}_{R,f1} & \mathbf{P}_{f1} & \mathbf{P}_{f1,f2} & \cdots & \mathbf{P}_{f1,fn} \\ \mathbf{P}_{R,f2} & \mathbf{P}_{f1,f2} & \mathbf{P}_{f2} & \cdots & \mathbf{P}_{f2,fn} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{R,fn} & \mathbf{P}_{f1,fn} & \mathbf{P}_{f2,fn} & \cdots & \mathbf{P}_{fn} \end{bmatrix} \tag{2.20}$$

where $\mathbf{x}_R$ is the position of the robot and $\mathbf{x}_{f1} \ldots \mathbf{x}_{fn}$ are the positions of feature. The values of the off-diagonal elements in the covariance matrix depend on the correlations between the state of the robot and features in the map.

While simple, the EKF-SLAM solution to the localization problem was proved inconsistent in [36], which showed that even in a stationary case with no process noise, the filter becomes inconsistent. Inconsistency occurs when the filter's estimated uncertainty does not match the true error. In extended Kalman filter (EKF) implementations, this is due to the sensitivity of the Jacobian matrix to noisy observations and erroneous estimates. Further, it was shown that this inconsistency cannot be circumvented by other nonlinear Kalman filtering methods, such as the UKF or IEKF [5], since they ultimately perform some level of linearization. It was, however, determined by Bailey, et al. [5] that inconsistency is fundamentally related to the "true" heading variance. If direct observations of the heading can be made, so that heading uncertainty is small, then the EKF-SLAM algorithm can at least avoid catastrophic failure, but not inconsistency [5]. The following subsections illustrate techniques which seek to further reduce linearization errors and computational complexities in order to enable full scale EKF SLAM.

2.8.1 Submapping Techniques.

Local sub-map techniques are a subset of the larger SLAM literature initially developed to reduce the computational cost of implementing full scale SLAM algorithms in real time. An added benefit is that using submaps can also mitigate global inconsistency. In local submapping algorithms such as [50],[51],[13], and [22],

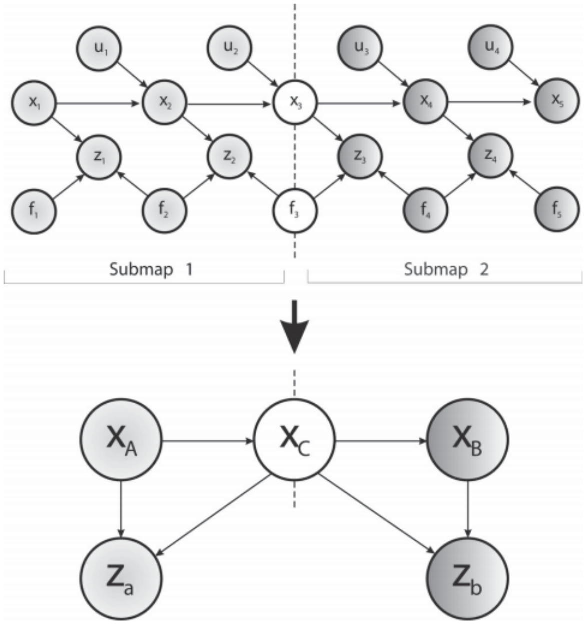


Figure 2.7: An Illustration of Supmapping. A Bayesian representation of submapping techniques shows that submaps are independent, given node X_c [51].

conditionally independent local submaps of bounded size or error are built and maintained in memory. The global map can then be reconstructed by piecing together these local maps. An illustration of the submapping problem is shown in Figure 2.7.

These maps, however, are populated by clusters of feature points such as corners, edges, and contours which consume large amounts of space and may not provide a meaningful reconstruction of the environment to the user. Furthermore, in order to close a loop, a re-observed feature must be matched to a feature stored in a submap. Considerable computation time can be consumed in order to find an observed feature buried in a submap.

2.8.2 Enforcing the Observability Conditions.

The fundamental flaw in EKF-SLAM is that the linearized system has an extra observable direction which is not observable to the true system [29]. This leads to inconsistency. To counter this problem, Hesch, et al. [29] propose a system which maintains the nullspace **N** at each timestep l by enforcing the conditions

$$\mathbf{N}_{l+1} = \Phi_l \mathbf{N}_l \ , \ \ \mathbf{H}_l \mathbf{N}_l = \mathbf{0}, \ \ l = 1, \ldots, k \tag{2.21}$$

through modifications to the state transition matrix,Φ_l, and the measurement Jacobian, $\mathbf{H}_l$. Enforcing these conditions prevents information gain in an unobservable direction and greatly improves consistency, at the cost of modifying Φ_l and $\mathbf{H}$ after linearization.

2.8.3 Inverse-Depth SLAM.

A disadvantage in monocular SLAM systems is the lack of depth information, particularly for features which exhibit low parallax – meaning that there is insufficient change in the features' positions when viewed in multiple frames. As opposed to classical monocular SLAM techniques, inverse-depth SLAM parametrizes each feature on initialization with six states [16]

$$\mathbf{y}_i = \begin{bmatrix} x_i & y_i & z_i & \theta_i & \phi_i & \rho_i \end{bmatrix}^T \tag{2.22}$$

where x_i, y_i, z_i are the coordinates of the camera's optical center; θ_i, ϕ_i are azimuth and elevation; and ρ_i is the inverse depth. Figure 2.8 illustrates how features are parametrized with inverse depth. Encoding features this way allows distant points to be tracked over time and to immediately contribute to the SLAM solution. If sufficient parallax occurs, the feature's depth uncertainty is reduced. While features at low parallax will take up six states, features at high parallax can be converted to the standard X, Y, Z representation to reduce computational burden. This parametrization, however, cannot eliminate linearization errors caused by the filter's dynamics or measurement equations [16].

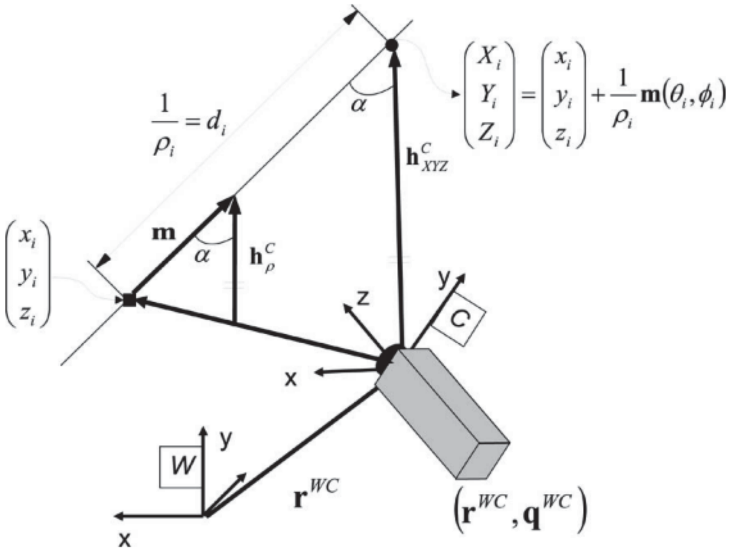

$\frac{1}{\rho_i} = d_i$
α
$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \frac{1}{\rho_i}\mathbf{m}(\theta_i, \phi_i)$
$\mathbf{h}^C_{XYZ}$
$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$
m
α
$\mathbf{h}^C_\rho$
z
y
C
x
W
y
x
$\mathbf{r}^{WC}$
$(\mathbf{r}^{WC}, \mathbf{q}^{WC})$

Figure 2.8: Inverse Depth Parametrization. An illustration of the inverse depth parametrization [16].

2.9 Pose SLAM

Another pillar in SLAM literature results from simply restructuring the vehicle's state [32] [38] [32] [41]. In the pose SLAM architecture, the state vector consists of a finite or continuous history of past camera poses. Features identified between camera poses are used to establish constraints between views. Feature information (or raw image data) is carried forward in time to establish relationships between all poses in a vehicle's trajectory, but the information is not typically included in the system state vector. Loop closures can then be treated more intuitively: a loop is closed when the trajectory intersects itself.

A key advantage in pose SLAM is the sparse nature of the information matrix which relates poses in the trajectory. The information matrix scales with the number of poses in the state vector and has non-null entries only for poses that are related [32]. This means that the information matrix is mostly tridiagonal (since consecutive poses are always related) with a small number of off-diagonal blocks corresponding to loop closures [32].

The formulation of the pose SLAM problem lends itself to non-linear optimization techniques such as bundle adjustment [21] [33] [65], which attempts to minimize reprojection error of features tracked over a finite or complete set of images with approximated camera poses. Bundle adjustment will be explained in greater detail in Section 2.10.4. Full bundle adjustment – using all camera poses and corresponding images – is typically not feasible as the primary estimator for an online navigation system, especially one with limited computational power. The following subsections outline methods for reducing complexity when using bundle adjustment to solve for an optimal trajectory.

2.9.1 Reduced Pose SLAM.

In general, not all poses or loop closures yield high information gain. This motivates the practice of ignoring redundant information so that only valuable information is retained and the sparsity of the information matrix is ensured. Konolige and Agrawal [38] proposed a reduced Pose SLAM system for indoor navigation based on skeleton frames taken at regular 5m intervals, which reduces the number of data points on a vehicle's trajectory.

Instead of including frame information at arbitrary intervals, Ila, et al. [32] showed that the information gain and distance between poses could be computed in closed form prior to deciding whether to include a pose in the trajectory. The distance **d** between two poses, x_i and x_n, can be estimated as Gaussian with mean $\bar{\mathbf{d}}$ and covariance $\mathbf{P}_d$

$$\begin{aligned} \bar{\mathbf{d}} &= h(x_i, x_n) \\ \mathbf{P}_d &= \begin{bmatrix} \mathbf{H}_i & \mathbf{H}_n \end{bmatrix} \begin{bmatrix} \mathbf{P}_{ii} & \mathbf{P}_{in} \\ \mathbf{P}_{in}^T & \mathbf{P}_{nn} \end{bmatrix} \begin{bmatrix} \mathbf{H}_i & \mathbf{H}_n \end{bmatrix}^T \end{aligned} \tag{2.23}$$

where $h(\cdot,\cdot)$ is the measurement equation and $\mathbf{H}$ is the measurement Jacobian, partitioned into the x_i portion ($\mathbf{H}_i$) and the x_n portion ($\mathbf{H}_n$). Ila, et al. choose to marginalize this multidimensional Gaussian for each of its dimensions, k, and determine the probability that the distance is within some interval $[-\kappa,\ \kappa]$

$$P(-\kappa \leq d_k \leq \kappa) = \int_{-\kappa}^{\kappa} \mathcal{N}(\bar{d}_k, \sigma_k^2) \tag{2.24}$$

If the probability is less than a defined threshold for all dimensions, the pose is included in the map [32].

For loop closures, the information gained is computed from the logarithm of the ratio of determinants of prior and posterior covariances [32], which simplifies to

$$\begin{aligned} \mathbf{S} &= \mathbf{R} + \begin{bmatrix} \mathbf{H}_i & \mathbf{H}_n \end{bmatrix} \begin{bmatrix} \mathbf{P}_{ii} & \mathbf{P}_{in} \\ \mathbf{P}_{in}^T & \mathbf{P}_{nn} \end{bmatrix} \begin{bmatrix} \mathbf{H}_i & \mathbf{H}_n \end{bmatrix}^T \\ \lambda &= \frac{1}{2} \ln \frac{|\mathbf{S}|}{|\mathbf{R}|} \end{aligned} \tag{2.25}$$

where $\mathbf{R}$ is the measurement covariance. If λ is above a certain threshold, there is sufficient information gain from the candidate loop closure and it should be included in the map.

2.9.2 Adaptive Windowing.

Another method for reducing complexity in the Pose SLAM framework is to limit the number of poses in the state. Rather than using a fixed window of poses, which risks eliminating valuable loop closure information, an adaptive windowing technique was introduced by Lim, et al. [41] that adapts the number of poses in the state vector to minimize information loss during loop closures and minimize complexity in open loop. Unfortunately, for long trajectories, it is impossible to avoid eliminating poses which close loops unless the window continues to grow.

2.10 Image Geometry

Image geometry draws insights from projective geometry. In the case of a single view, it will be shown that given coordinate information about an object in a scene, the extrinsic parameters of the camera viewing the object (with respect to the object's coordinate frame) can be extracted. In the case of multiple views, it will be shown that the relative geometry between frames can be exploited to determine the motion of the sensor, or sensors, which took the images.

2.10.1 Single View Geometry .

In Dean's work [18] the calculation of vanishing points was supported by the known structure of indoor environments. This information was built into the system – i.e. the system had no way of recognizing it operated in a specific environment. In a similar way, the perspective n-point (PnP) problem relates known environment information to image data. Let u and v be the pixel locations for a point $[x\ y\ z]^T$ in 3-space. Pixel coordinates are related to a three dimensional point by the equations [58]

$$\begin{aligned} u &= \frac{f_x(r_{11}x + r_{12}y + r_{13}z + t_x) + \gamma(r_{21}x + r_{22}y + r_{23}z + t_y)}{r_{31}x + r_{32}y + r_{33}z + t_z} + c_x \\ v &= f_y \frac{r_{21}x + r_{22}y + r_{23}z + t_y}{r_{31}x + r_{32}y + r_{33}z + t_z} + c_y \end{aligned} \tag{2.26}$$

where each r_{ij} is an element of the rotational matrix $\mathbf{R}$ and $[t_x\ t_y\ t_z]^T$ is the translation from the frame of the imaged points to the camera frame. The parameters c_x, c_y, f_x, f_y, and γ are known from camera calibration.

In the PnP problem, both the 3-D points and the pixel locations are known. Since each pair (u, v) contributes two equations for 12 unknowns, 6 known points are needed to find a solution. By realizing that the rotational matrix can be decomposed into three angular parameters, the number of unknowns becomes 6 and only 3 known points are needed, as shown in [26]. However, factoring in uncertainty and nonlinearities, some PnP algorithms involve at least 4 known points [39] [27] [30].

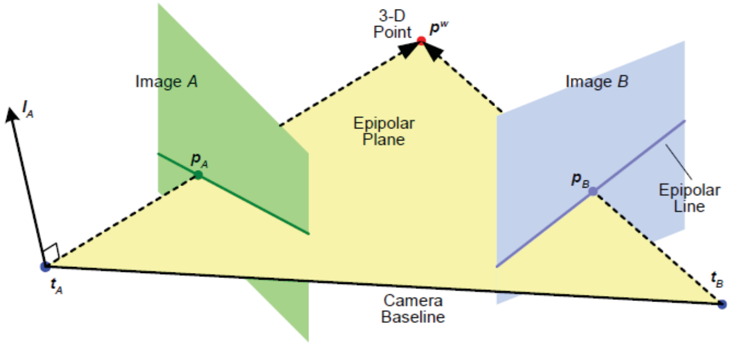


Figure 2.9: An Illustration of the Epipolar Constraint. Two camera centers viewing the same 3D point lie on an epipolar plane. Under pure translation, the point in the image will move along the epipolar line [62].

2.10.2 *Two Views .*

Epipolar geometry describes the relationship between two views of a scene. The study of epipolar geometry is typically motivated by finding corresponding points in stereo images [28]. Let the coordinates of a point $\mathbf{p}^w$ in a world coordinate frame be imaged in two views as the points $\mathbf{x}$ and $\mathbf{x}'$. The points $\mathbf{x}$, $\mathbf{x}'$ are the projections of the pointing vectors $\mathbf{p}_A, \mathbf{p}_B$, respectively, onto the images A and B. Epipolar geometry states that these pointing vectors, the point $\mathbf{p}^w$, and the camera centers $\mathbf{t}_A, \mathbf{t}_B$ are all coplanar, as shown in Figure 2.9.

For two calibrated cameras, the normalized points $\hat{\mathbf{x}}'$ and $\hat{\mathbf{x}}$ are related up to scale through the essential matrix

$$\begin{aligned} \hat{\mathbf{x}}'^{\mathrm{T}}\mathbf{E}\hat{\mathbf{x}} &= 0 \\ \mathbf{E} &= [\mathbf{t}]_{\times}\mathbf{R} \end{aligned} \tag{2.27}$$

The decomposition of E into $[\mathbf{t}]_\times\mathbf{R}$ means that rotation and unscaled translation can be extracted from the essential matrix relating two views of a scene.

Much like the equations for a single view, the elements of the essential matrix can be formed into a vector and expressed by the system of linear equations

$$\begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} \mathbf{e} = \mathbf{0} \tag{2.28}$$

As outlined in [28], this system can be solved by RANSAC with only 7 point correspondences. This solution is then further optimized to yield a final estimate of the essential matrix.

After the essential matrix is recovered, the camera matrices representing the rotation matrix **R** and unscaled translation vector **t** can be recovered by singular value decomposition [28]. First we define the normalized camera matrices as

$$\mathbf{P} = [\mathbf{I} \mid \mathbf{0}], \quad \mathbf{P}' = [\mathbf{R} \mid \mathbf{t}] \tag{2.29}$$

Letting E have SVD

$$\mathbf{E} = \mathbf{U\Sigma V}^T$$
$$\mathbf{\Sigma} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.30}$$

Hartley and Zisserman show that one of the solutions for **P′**, corresponding to the reconstructed points being in front of both cameras is

$$\mathbf{P}' = [\mathbf{U}\mathbf{W}^{\mathrm{T}}\mathbf{V}^{\mathrm{T}} \mid \mathbf{u}_3]$$
$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.31}$$

where $\mathbf{u}_3$ is the third column of $\mathbf{U}$. Note that a property of the essential matrix is that it is a singular matrix and its two non-zero singular values are identical.

Epipolar geometry was used in [20] to constrain the motion between two consecutive images to the epipolar plane. In other words, by definition of the epipolar plane, the baseline between two cameras viewing the same point must be contained in the epipolar plane, or mathematically [20]

$$\Delta\mathbf{x} = \mathbf{t}_b - \mathbf{t}_a$$
$$\Delta\mathbf{x} \cdot (\mathbf{p}_a \times \mathbf{p}_b) = \mathbf{0} \tag{2.32}$$

Diel, et al. [20] propose a residual based on Equation 2.32 that is a projection of out-of-plane translation onto the epipolar plane. The measurement equation for the residual is

$$\mathbf{e}_z = \frac{\mathbf{p}_a \times \mathbf{p}_b}{||\mathbf{p}_a \times \mathbf{p}_b||} \quad \mathbf{e}_x = \frac{\Delta\mathbf{x}}{||\Delta\mathbf{x}||}$$
$$\hat{\mathbf{x}} = (\mathbf{I} - \mathbf{e}_x\mathbf{e}_x^T)\mathbf{e}_z\mathbf{e}_z^T\Delta\mathbf{x} \tag{2.33}$$

which vanishes when the epipolar constraint is satisfied. In this framework, each individual feature matched between two frames incrementally rotates the camera baseline to a position which minimizes the residual error.

2.10.3 Three Views.

Three-view geometry is a natural extension of two views. A three-view constraint involves finding two-view constraints between three images that are consistent with each other. Three-view constraints were used by Indelman, et al. [35] in a vision-aided navigation system, illustrated in Figure 2.10. Using the notation in Figure 2.10, the translations $\mathbf{T}_{ij}$ must satisfy the constraints [35]

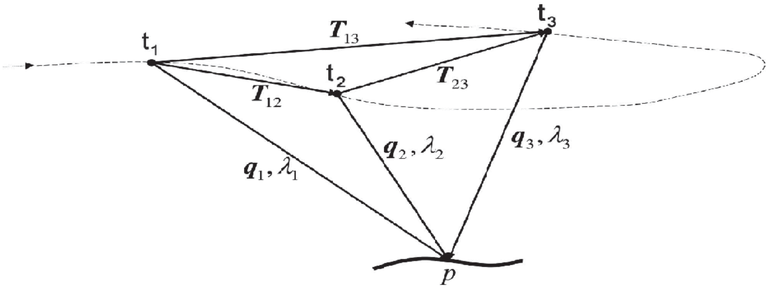


Figure 2.10: Three Views of a Common Point. Three views of the same feature can be used to establish a three-view constraint on the three cameras which viewed the feature [35].

$$\begin{aligned} \mathbf{q}_1^T(\mathbf{T}_{12} \times \mathbf{q}_2) &= 0 \\ \mathbf{q}_2^T(\mathbf{T}_{23} \times \mathbf{q}_3) &= 0 \\ (\mathbf{q}_2 \times \mathbf{q}_1)^T(\mathbf{q}_3 \times \mathbf{T}_{23}) &= (\mathbf{q}_1 \times \mathbf{T}_{12})^T(\mathbf{q}_3 \times \mathbf{q}_2) \end{aligned} \tag{2.34}$$

which are simply two epipolar constraints and a third constraint that forces consistency between the three translations.

The algorithm begins by extracting feature matches from three overlapping images using SIFT. Matching is first performed between the images 1 and 2, and 2 and 3. In order to robustify these matches, the fundamental matrix is computed in a RANSAC approach for both image pairs. Matching is then performed between those two sets to determine triplet matches . The set of triplet matches are then used in an IEKF to aid an IMU.

2.10.4 n Views and Bundle Adjustment .

Bundle adjustment, as its name would suggest, is the process of minimizing the re-projection error over a set of image points in *n* views by adjusting the three dimensional coordinates of each imaged point, the camera calibration parameters, and the associated camera pose (which form a bundle of rays). An illustration of the bundle adjustment problem for a single frame is shown in Figure 2.11.

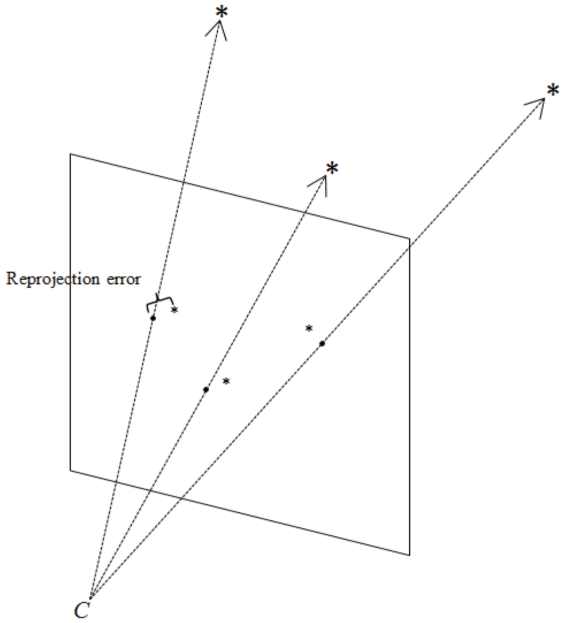


Figure 2.11: An Illustration of the Bundle Adjustment Problem. The goal of bundle adjustment is to iteratively refine the camera position and 3D feature locations so that the rays (shown as arrows) pass through the imaged points of each 3D feature.

The goal is to find a vector $\mathbf{x}$, consisting of all camera poses $\hat{\mathbf{P}}$ and calibration parameters $\mathbf{K}$, and 3D feature locations $\hat{\mathbf{X}}$, which maximizes a log-likelihood function. To begin, let $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\}$ be independent measurements assumed to be from independent Gaussian pdf's with means $\mathbf{\Theta}_i$ and identical covariances $\mathbf{\Sigma}$. The measurements are related to each mean by the function $\mathbf{\Theta}_i = \mathrm{h}(\mathbf{K}, \hat{\mathbf{P}}, \hat{\mathbf{X}})$. The likelihood function $L(\mathbf{\Theta}|\mathbf{Z})$ to maximize, where $\mathbf{\Theta} = \{\mathbf{\Theta}_1, \mathbf{\Theta}_2, \ldots, \mathbf{\Theta}_n\}$, is [12]

$$\begin{aligned}
L(\mathbf{\Theta}|\mathbf{Z}) &\propto \prod_{i=1}^{n} \exp\Big((-1/2)[\mathbf{\Theta}_i - \mathbf{z}_i]^T \mathbf{\Sigma}^{-1} [\mathbf{\Theta}_i - \mathbf{z}_i]\Big) \\
&= \exp\left((-1/2) \sum_{i=1}^{n} [\mathbf{\Theta}_i - \mathbf{z}_i]^T \mathbf{\Sigma}^{-1} [\mathbf{\Theta}_i - \mathbf{z}_i]\right) \\
\log(L(\mathbf{\Theta}) &\propto (-1/2) \sum_{i=1}^{n} [\mathbf{\Theta}_i - \mathbf{z}_i]^T \mathbf{\Sigma}^{-1} [\mathbf{\Theta}_i - \mathbf{z}_i] \\
&= (-1/2) \sum_{i=1}^{n} ||\mathbf{\Theta}_i - \mathbf{z}_i||^2_{\mathbf{M}} \qquad (2.35)
\end{aligned}$$

where $||\mathbf{\Theta}_i - \mathbf{z}_i||^2_{\mathbf{M}}$ is the Mahalanobis distance [28]. Maximizing the log-likelihood function is equivalent to finding a $\mathbf{\Theta}$ such that

$$\mathbf{\Theta} = \arg \min_{\mathbf{\Theta}} (-1/2) \sum_{i=1}^{n} ||\mathbf{\Theta}_i - \mathbf{z}_{\mathbf{i}}||^2_{\mathbf{M}} \qquad (2.36)$$

There are many techniques for solving this system, the most basic of which being weighted least squares [65]. It's important to note the large size of the state vector being optimized in the vision navigation case. This motivates techniques for efficiently solving the least squares problem which take advantage of the sparse nature of the Jacobian matrix [21] [33] [65] [37].

2.11 Stereo Vision SLAM

Another popular approach in SLAM literature involves measuring the three-dimensional coordinates of feature points with a stereo vision system (SVS) [23] [17] [57] [15]. In a SVS, two cameras are rigidly mounted to a vehicle and their relative position – the camera baseline – and orientation is calibrated. Based on epipolar geometry, as described in Section 2.10.2, the image coordinates of each feature identified as common to both cameras in the SVS can be used with the calibrated camera parameters to determine the three-dimensional location of a feature. In a SLAM network, this three-dimensional feature measurement can be used to simultaneously update the coordinates of its corresponding feature in the map and the location of the vehicle.

The accuracy of the SVS is determined by the error in the calibrated internal and external parameters. In [67] it was shown that errors in calibration for the external parameters, the relative translation and rotation between the stereo camera pair, directly affect reconstruction accuracy and are the dominating factors for determining the location of an object in three-dimensions. Further, it can be shown that reconstruction accuracy increases as the camera baseline extends. Thus, when using a SVS on a single vehicle the maximum reconstruction accuracy is limited by the dimensions of the vehicle which determines where the cameras can be mounted.

2.12 Constraint Based Localization

As an alternative to SLAM, feature correspondences over two or more images can be used to set up constraints on camera motion. The features are not included in the state and therefore the solution involves less computation.

The multi-state constraint Kalman filter developed by Mourikis and Roumeliotis [48] uses feature correspondences across multiple images to set up constraints on camera motion and IMU estimates in an EKF. Each time a new image is captured, the state vector is augmented with states representing the estimated camera position and attitude such that at any time the state vector contains a series of N camera poses and the state of the IMU. Each feature measurement imposes a constraint on the camera pose and feature location. Mourikis and Roumeliotis form the residual vector [48]

$$\mathbf{r} = \mathbf{z} - \hat{\mathbf{z}} \tag{2.37}$$

$$\mathbf{r} = \mathbf{H}_X\mathbf{X} + \mathbf{H}_f\mathbf{p}_f + \mathbf{n} \tag{2.38}$$

where $\mathbf{H}_X$ and $\mathbf{H}_f$ are the Jacobians of the measurement $\mathbf{z}$ with respect to the state and feature positions respectively, $\mathbf{X}$ is the state, $\mathbf{p}_f$ is the error in the position estimate of the feature, and $\mathbf{n}$ is white Gaussian noise. By projecting the residual onto the left nullspace of $\mathbf{H}_f$, the residual becomes independent of the errors $\mathbf{p}_f$ and EKF updates can be performed without including the feature in the filter state [48].

2.13 Hybrid Methods

The last few sections outlined two of the broad techniques for localizing a vehicle: SLAM and nonlinear optimization using multi-view constraints. This section outlines works which synthesize these methods. These hybrid methods take advantage of the computational performance of SLAM and the high accuracy of nonlinear optimization. While Pose SLAM traditionally allows for bundle adjustment following completion of a trajectory, these methods implement bundle adjustment as a background process which is able to integrate its solution into the active filtering process.

Perhaps the pioneer work in this type of method is the dual-layer estimator proposed by Mourikis and Roumeliotis [49]. This work implements their multi-state constraint Kalman filter as the primary estimator and uses a batch estimator to close loops. Candidate loop closures are chosen by evaluating a distance criterion between camera poses. The candidates are then processed with image and inertial information from memory by a bundle adjustment algorithm running in parallel. Reference [49] does not, however, succinctly explain how the bundle adjustment solution is fed back to the active filter.

Kaess, et al. [37] showed that representing the information matrix of the pose SLAM problem in a graphical way more easily shows that smoothing and mapping can be implemented as separate processes as shown in Figure 2.12. Given the separator, the operations of smoothing and mapping are clearly independent processes.

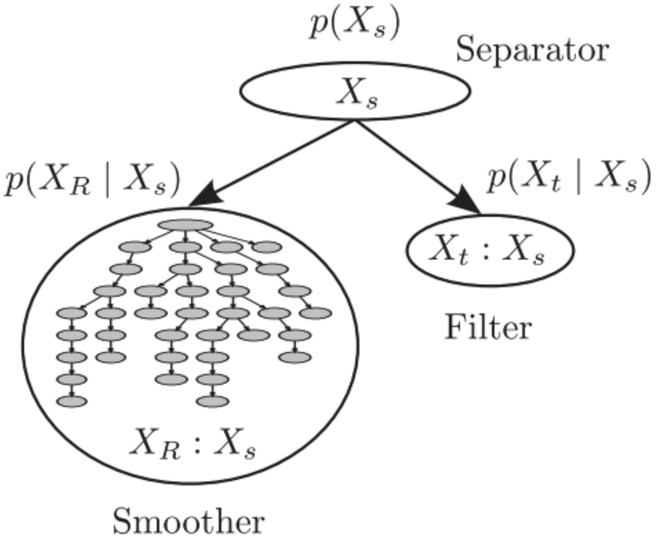


Figure 2.12: Separation of the Smoothing and Mapping Processes. The bayes tree representation of the information matrix shows that smoothing and mapping can be performed separately [37]

The algorithm, named incremental smoothing and mapping (iSAM), also has the advantage of incrementally solving the nonlinear system. All past poses marginalized into the smoother are factored into ”cliques” so that adding a new pose only affects a smaller subset of poses related to the new pose. This attribute helps alleviate data latency and improves the overall estimation framework.

2.14 Extension to Multiple Vehicles

Localization performed by multiple vehicles has been shown to have a substantial improvement over single vehicle localization [47] [53]. Even a vehicle with highly accurate sensors can benefit from measurements made by another vehicle with less accuracy.

One framework for multiple vehicle localization is to navigate in an alternating fashion where one section of the vehicles fix their position and act as landmarks for the rest of the group to navigate [61] [19]. For vision navigation, viewing vehicles with known geometry can remove some scale ambiguity, resulting in a more accurate navigation solution. This ”leap-frog” approach is suitable for ground vehicles which can easily act as stationary landmarks, but is far less applicable to aerial vehicles which are difficult to hold in place without landing.

A more appropriate framework for a network of aerial vehicles is for all vehicles to move continuously while identifying matching points in their views. In [45][34], passive measurements of another vehicle are extracted from three overlapping views of the same scene. By identifying features common to each image, these systems simultaneously solve for the optimal camera orientations and feature locations in all three images. This requires, however, that each new image be compared to all images in a repository in real time to determine if two images of the same scene exist.

2.15 Autonomous Navigation

Up to this point, only the topic of localization and mapping has been discussed. The logical next step from a reliable navigation system is to incorporate autonomous command and control, as well as autonomous coordination, synchronization, and consensus in the case of multiple vehicles.

At a fundamental level, vehicle autonomy must achieve active decision making with respect to physical cues. For example, identification of the end of a hallway might be used in planning a path for exploration. In this sense, decision making is often reactive which implies that autonomy can be achieved by commanding or training a response to specific inputs. Autonomy is often highly dependent on navigation. Autonomous decisions are frequently tied to a navigational cue, such as position, velocity, or orientation.

As an example, consider the autonomous navigation of [3]. In this work, Adler and Xaio propose an unmanned system capable of mapping unknown urban environments through autonomous path planning and multiple sensor fusion. While the mapping algorithm used in this system is made simple through the fusion of a GNSS antenna and IMU, the point remains that the algorithm must have some sense of its location. Without such information, the system would be unable to determine which areas had already been explored.

2.16 Previous Work

In the previous work [18], Dean developed a real-time navigation system to enhance the position and orientation accuracies of an ARDrone quad-rotor helicopter. Through experimentation, it was discovered that the heading reported by the ARDrone tended to randomly drift during runtime. To correct this drift, the navigation system tracked perspective features called vanishing points, which are the theoretical intersection of parallel lines at infinity. Because these vanishing points exhibit little to no parallax as the vehicle moves, they serve as absolute references for determining heading. An EKF was then used to fuse the measurements reported by the ARDrone with the vanishing point computed from processing a frame of the video stream.

2.16.1 System Design and Dynamics.

The Kalman filtering system of Capt Dean's design was implemented in off-board software which was integrated into a navigation graphical user interface (GUI). The GUI was provided in a software development kit (SDK) released by Parrot SA [1]. The GUI provides live video, displays telemetry data, and enables joystick control. The developed navigation software intercepts a video frame after it is decoded by one of the SDK threads. Once decoded and if the vanishing point algorithm is ready, the image is processed by the algorithm to predict the vanishing point. The predicted vanishing point is passed to a command thread which computes a control input to correct heading errors. A block diagram of the system is presented in Figure 2.13.

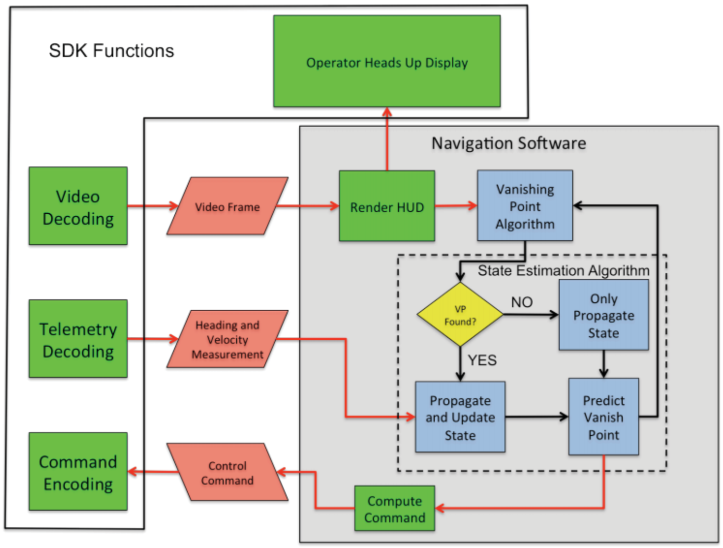

SDK Functions
Operator Heads Up Display
Navigation Software
Video Decoding
Video Frame
Render HUD
Vanishing Point Algorithm
State Estimation Algorithm
VP Found?
NO
Only Propagate State
YES
Telemetry Decoding
Heading and Velocity Measurement
Propagate and Update State
Predict Vanish Point
Command Encoding
Control Command
Compute Command

Figure 2.13: System Design in the Previous Work. The diagram of the system design in [18] shows the relationship between the developed software and the provided SDK.

Due to limited access to the ARDrone's hardware and visual odometry, the stochastic system model used in [18] had to take a simplified form. Due to the frequency of measurements, the input to the system was neglected so that the simplified stochastic model of the system took the form

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{G}(t)\mathbf{w}(t) \tag{2.39}$$

so that the system dynamics were assumed linear between measurements. The state vector $\mathbf{x}(t)$ was composed as

$$\mathbf{x} = \begin{bmatrix} X^N & Y^N & \delta\psi & \delta\dot{\psi} & V_x^N & V_y^N \end{bmatrix}^T \tag{2.40}$$

where X^N, Y^N were the horizontal position of the drone in the navigation (hallway) frame, $\delta\psi, \delta\dot{\psi}$ were the heading error and heading error rate of the drone-reported heading, and V_x^N, V_y^N were the velocities of the drone in the navigation frame.

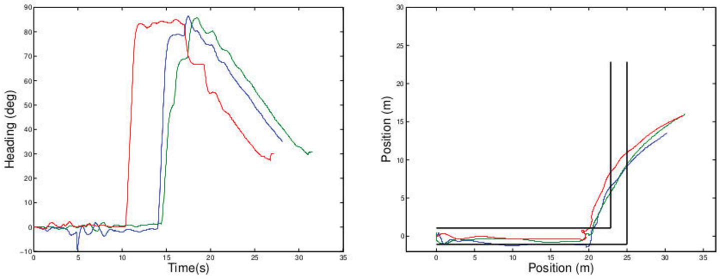
One of the core algorithms running in the software is the vanishing point estimator. The vanishing point estimation algorithm begins when a new frame becomes available. The frame is processed to remove distortion and extract lines. If a predicted vanishing point exists, only lines within five degrees of the predicted vanishing point are considered. Otherwise, all lines are considered candidates [18].

The vanishing point algorithm uses a RANSAC algorithm to estimate the vanishing point from candidate lines. This point is converted to a unit vector and passed to the state estimator, which uses the vector and velocities reported by the drone to compute a measurement update. The heading correction from the measurement update is converted into a control input and encoded for execution by the drone.

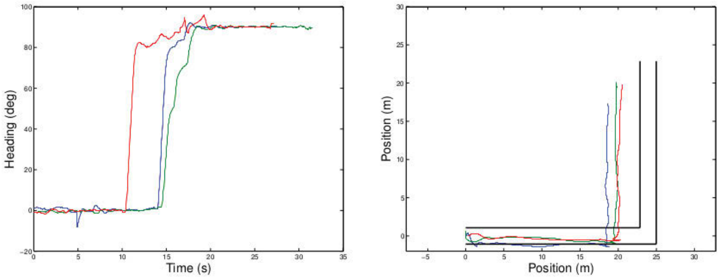
2.16.2 Current System.

The system's primary weak-point is in position estimation. Prior to testing, strips of tape needed to be added to the test course so that the drone's velocity estimation algorithm could function. Dean's system uses the vanishing point algorithm to rotate velocity measurements reported by the drone into the navigation frame. Velocities are then integrated into position estimates through an EKF update. Figure 2.14 shows the results of Capt Dean's software.

The effect of of the velocity measurements is evident from the estimated ground track. The velocity measurements are highly dependent on environment texture and are often inaccurate, which ultimately leads to error in the estimated position. Furthermore, the fact that texture needed to be added to the environment for the drone's velocity



(a) Reported Heading (b) Reconstructed Ground Track



(c) Estimated Heading (d) Estimated Ground Track

Figure 2.14: Results of the Previous Work. The results of [18] show a great improvement from the heading reported by the drone (a) and the heading estimated by the filter (b). However, the inaccuracies in the velocity measurements causes an error in the vehicle position.

estimation algorithm to function reduces the system's capability for autonomous navigation. Much like Capt Dean's work, the system described in the following chapters seeks a more accurate navigation solution through image processing techniques, but without having to alter the navigated environment. The system also gains the advantage of additional information from a second drone operating in its field of view.

2.17 Summary

This chapter has provided the context for the rest of this thesis by highlighting major areas in the field of vision-aided navigation. The next chapter will draw upon the foundations of this chapter to improve the navigation system of [18] and extend its operation to a cooperative network of vehicles.

III. Methodology

The goal of this research effort is to develop a system which is able to process data from multiple vehicles to accurately determine the full three-dimensional position of all vehicles in the network. This chapter will discuss the architecture and algorithms used in the multiple vehicle system.

3.1 System Architecture

The system designed in this chapter consists of a single laptop running centralized navigation software for two independently operating ARDrone quad-rotor helicopters. Both ARDrones are able to communicate with the ground station laptop through a wireless router which is connected to the laptop through its Ethernet port. The complete system setup is shown in Figure 3.1.

Some of the software developed takes advantage of an open-source meta-operating system called ROS [52]. ROS provides many advantages over other robotics software, such as hardware abstraction, low-level device control, and package management, and it is designed to integrate seamlessly with different hardware, programming languages, and the like [52].

The main building blocks in ROS are nodes – executable packages of code which perform some process or processes. Nodes typically communicate with each other by passing messages through topics. A node can either send messages through a topic (publish) or receive messages from a topic (subscribe). In this way nodes are pieced together into a single executable program through their common topics. A more detailed description of ROS can be found at [2].

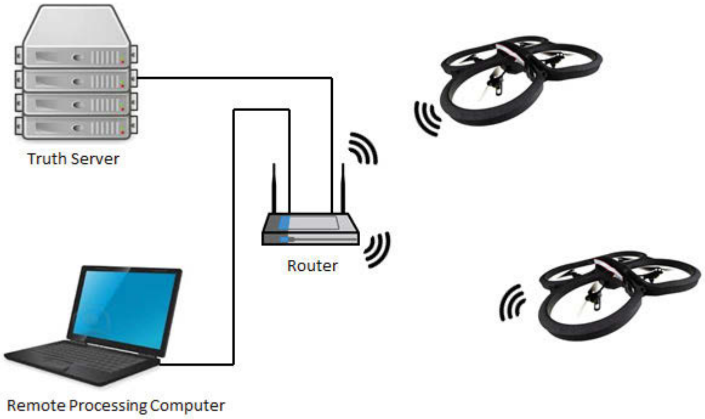


Figure 3.1: System Setup. The system setup consists of two independent vehicles communicating with a central processing computer through a WiFi connection. The truth server (optional) is also connected through the router.

3.1.1 The ARDrone Platform.

The ARDrone by Parrot SA is a small commercial grade quadrotor vehicle. The vehicle has a forward looking high-definition camera, a downward looking camera, an ultrasonic altimeter, a barometric altimeter, a magnetometer, and a MEMs grade IMU. Inertial, altimeter, and image information is fused in the vehicle's on-board microprocessor to produce an estimate of attitude and altitude. The downward looking camera is used to measure the vehicle's velocity using optical flow. The telemetry data is combined with the compressed video streams and status messages and broadcast over a wireless link [1].

An SDK for a desktop computer is also provided by Parrot SA to give developers access to video and telemetry data [1]. In addition, the manufacturer does not support compiling software on the ARDrone's on-board processor. Therefore, all algorithms developed in this research were implemented on a laptop.

3.1.2 Hardware and Software.

The computer used to develop the navigation software is a HP EliteBook 8560w laptop. The computer has an Intel Core i7 quad-core processor running at 2.2 GHz, 16 gigabytes of RAM, and a 256 gigabyte solid state disk drive. The operating system used is Ubuntu Linux 12.04, running version 3.20-35 of the Linux kernel. The navigation software uses the computer vision library OpenCV version 2.4.2, the ROS Groovy distribution, and MATLAB version R2012a.

3.1.2.1 ROS Packages.

A number of packages used in this system are available on-line. These packages include: the ROS ARDrone driver, developed in the Autonomy Lab of Simon Fraser University [46]; the Vicon bridge package; the joystick driver package; and the ROS camera calibration package.

The ROS ARDrone driver wraps around the official ARDrone SDK version 2.0 [1] to enable communication with the ARDrone in standard ROS style – through messages, parameters, and services. All data streams decoded by the SDK, including video and navigation data, are converted into separate ROS topics. Likewise, all parameters specific to the ARDrone, such as control gains, are loaded into the ROS parameter server under the driver node's namespace. In addition, the ARDrone driver subscribes to a generic command topic which allows greater flexibility in the device used for manual control.

3.1.2.2 Multiple Vehicle Network.

Built into the ROS ARDrone driver is the capability to specify an IP address of the ARDrone controlled during runtime. This capability enables control of multiple vehicles by simply running two instances of the ARDrone driver with separate IP addresses and forcing connection to a central network.

On start-up, each ARDrone is automatically configured to have a pre-specified IP address. To change the IP and force the ARDrone to connect to a managed network, an executable file was written in its memory which disables the old wireless link, configures the new IP, and forces connection to the router network. Once the executable file is run, each ARDrone is accessible through the router.

3.1.3 Concept Diagram.

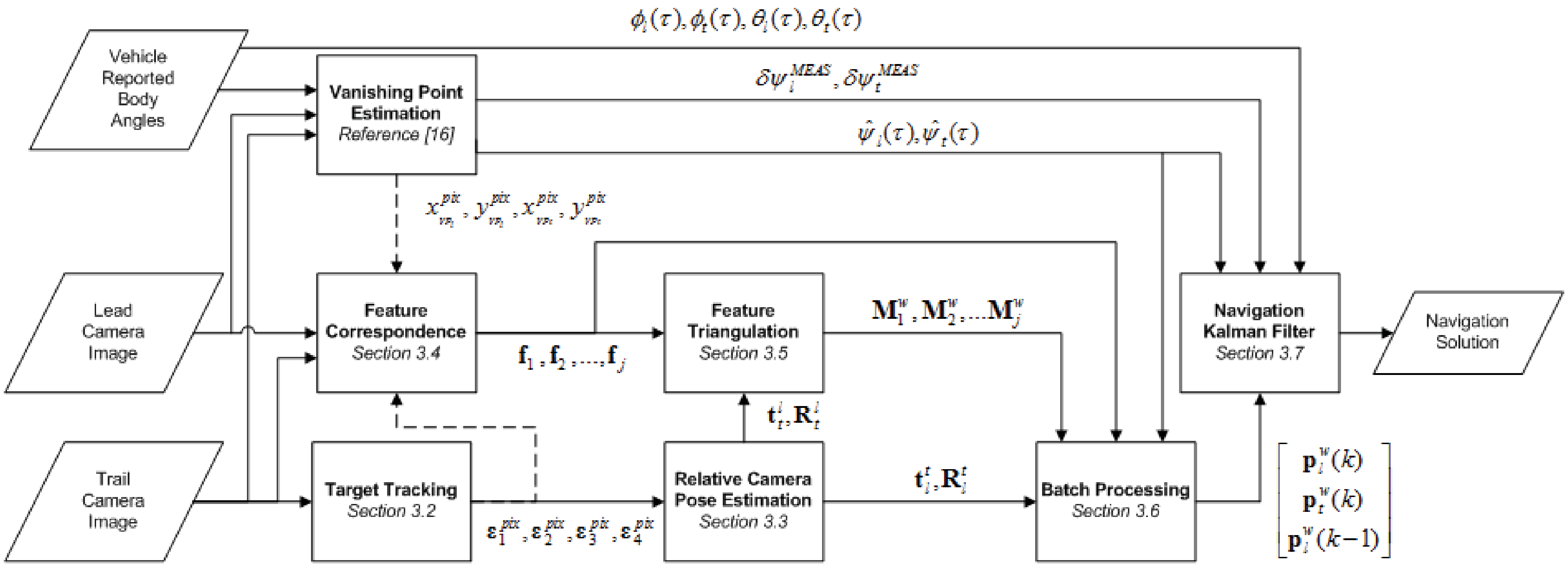
Figure 3.2 outlines the conceptual flow of the algorithms in this thesis. Each block in the concept diagram represents an algorithm which takes specific inputs and transforms them into an output. The final output in the concept diagram is the navigation solution, which estimates the current state of both vehicles. The remainder of this chapter will describe each block, its inputs, and its outputs in detail.

3.2 Target Tracking

The goal of target tracking is to estimate the image coordinates of each of the four targets $\{\varepsilon_1^{pix}, \varepsilon_2^{pix}, \varepsilon_3^{pix}, \varepsilon_4^{pix}\}$ from a trail camera image. The target locations will be used to estimate relative camera pose and aid in feature correspondence. Target tracking consists of identifying a bounding pixel region which contains a target, determining the pixels which belong to the target, and approximating the center of the target. Target tracking assumes that the amount of motion between images is small and such that the bounding region will contain the target's new location in the next frame.

3.2.1 Vehicle Targets.

The vehicle targets used in the target tracking algorithm are ultra-bright LEDs housed inside colored table tennis balls. The table tennis ball material acts as a diffuser to convert a uni-directional LED into an omni-directional light source. The fashioned targets are shown in Figure 3.3. The high intensity of the markers make them easier to identify in an image, while also reducing the algorithm's sensitivity to ambient lighting.



Vehicle Reported Body Angles
Lead Camera Image
Trail Camera Image
Vanishing Point Estimation
Reference [16]
Feature Correspondence
Section 3.4
Target Tracking
Section 3.2
Feature Triangulation
Section 3.5
Relative Camera Pose Estimation
Section 3.3
Batch Processing
Section 3.6
Navigation Kalman Filter
Section 3.7
Navigation Solution
$\phi_l(\tau), \phi_t(\tau), \theta_l(\tau), \theta_t(\tau)$
$\delta\psi_l^{MEAS}, \delta\psi_t^{MEAS}$
$\hat{\psi}_l(\tau), \hat{\psi}_t(\tau)$
$x_{vp_1}^{pix}, y_{vp_1}^{pix}, x_{vp_2}^{pix}, y_{vp_2}^{pix}$
$\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_j$
$\mathbf{M}_1^w, \mathbf{M}_2^w, \ldots \mathbf{M}_j^w$
$\mathbf{t}_t^l, \mathbf{R}_t^l$
$\mathbf{t}_l^t, \mathbf{R}_l^t$
$\varepsilon_1^{pix}, \varepsilon_2^{pix}, \varepsilon_3^{pix}, \varepsilon_4^{pix}$
$\begin{bmatrix} \mathbf{p}_l^w(k) \\ \mathbf{p}_t^w(k) \\ \mathbf{p}_l^w(k-1) \end{bmatrix}$

Figure 3.2: Concept Diagram.

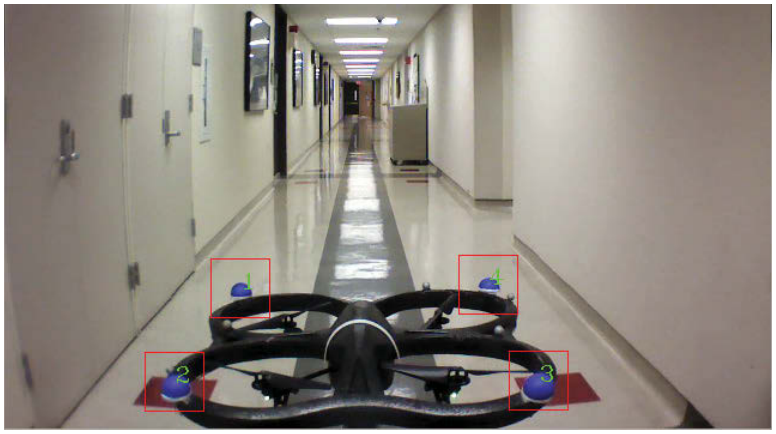


Figure 3.3: Tracking Vehicle Targets. The LED markers on the leading ARDrone vehicle are easily extracted from the surrounding image. The locations of each target are found by determining the pixels within the color range of a target and finding its central spatial moment.

3.2.2 Tracking Loops.

Each target on the lead vehicle is given its own square region of interest (ROI), as can be seen in Figure 3.3. The ROI is made small enough to cut out as much of the surrounding image as possible, but large enough to allow for movement of the target between frames.

Each bounding ROI can be initialized from a ROS service [2] which prompts the user to left-click each target in the correct order. The order of the targets must correspond with the order of the targets' world frame coordinates in memory: starting from the top left target and proceeding counter-clockwise. The bounding ROI is computed for the next frame based on the target area in the current frame. As targets move farther away, the target area location becomes more sensitive to changes in camera orientation. Therefore, the computed ROI is proportional to the inverse area – as the target gets smaller, the ROI becomes larger.

For each target, its center ε_i^{pix} is found by processing its corresponding ROI. First, each pixel in the image region is converted from RGB to HSV encoding. HSV encoding represents a pixel's color by a three dimensional vector: hue (H); saturation (S); and value (V), or brightness. Next, the pixels which belong to the target are determined by their color. Finally, the center of the target is found by determining the center of mass of the blob of pixels which belong to the target.

3.3 Relative Camera Pose Estimation

Relative camera pose estimation is used to determine the translation $\mathbf{t}_l^t$ and orientation $\mathbf{R}_l^t$ between the leading and trailing vehicles' cameras from the image coordinates of each target $\{\varepsilon_1^{pix}, \varepsilon_2^{pix}, \varepsilon_3^{pix}, \varepsilon_4^{pix}\}$, the measured coordinates of each target in the lead camera frame, and the calibration parameters for the trail camera. The vector $\mathbf{t}_t^l$ represents the relative position from the origin of the trail camera frame to the origin of the lead camera, expressed in the trail camera frame. The matrix $\mathbf{R}_t^l$ is a DCM which transforms coordinates in the trail camera frame to the lead camera frame.

The locations of each target in the trail vehicle's image are matched up to the true target locations expressed in the lead vehicle's camera frame, which have been determined ahead of time. Each imaged target ε_i^{pix}, true target location, and the calibration parameters of the trail camera can be used to form the equations shown in Section 2.10.1, where the x and y pixel locations of the imaged target correspond to the variables u and v respectively. The equations for the four targets form a overdetermined PnP problem which can be solved using non-linear techniques.

The non-linear solver used in this work is OpenCV's `solvePnP` function. The `solvePnP` function can be configured to run in three modes of operation: `CV_P3P`, which runs the algorithm in [26]; `CV_EPNP`, which runs the algorithm found in [39]; or `CV_ITERATIVE`, which is a custom Levenburg-Marquardt optimization routine. The function can also be used in a custom RANSAC mode, where any of the above three methods can be used as the PnP solver [10]. For this work, the `CV_ITERATIVE` mode will be used.

3.4 Feature Correspondence

The goal of feature correspondence is to find a set of vectors $\mathbf{f}_j$ which represent the pixel coordinates corresponding to a single feature identified in four images. The j-th feature vector can be written as

$$\mathbf{f}_j = \begin{bmatrix} \mathbf{x}_{lj}^{pix}(k) & \mathbf{x}_{tj}^{pix}(k) & \mathbf{x}_{lj}^{pix}(k-1) & \mathbf{x}_{tj}^{pix}(k-1) \end{bmatrix} \tag{3.1}$$

where l and t denote the lead and trail pixel frames, k is the current time epoch, and $k-1$ is the previous time epoch. Feature correspondence involves feature extraction (i.e. detection), description, and matching, as detailed in Section 2.5.

Initial feature correspondence determines matches between the lead and trail camera images at the same time epoch. It is performed by OpenCV's feature detector, descriptor extractor, and descriptor matcher objects. OpenCV offers a number of traditional methods, such as SIFT [43] and SURF [6], as well as some of the newly developed feature detection algorithms, such as ORB [55], and binary descriptor extractors, such as BRISK [40] and FREAK [4]. The matching algorithms include: brute force, where matches are determined strictly by descriptor distance; and k-nearest-neighbor, where each descriptor may have up to k matches. The detector, descriptor, and matcher combination used in this research will be discussed in Chapter 4.

After initial feature correspondence, the raw matches are filtered using an estimate of the fundamental matrix $\mathbf{F}$, computed by the equation

$$\mathbf{F} = (\mathbf{T}_{c_l}^{pix})^{-T}\mathbf{E}(\mathbf{T}_{c_t}^{pix})^{-1} \tag{3.2}$$

where $\mathbf{T}_{c_l}^{pix}$ and $\mathbf{T}_{c_t}^{pix}$ are the camera calibration matrices for the lead and trail vehicles, respectively, and $\mathbf{E}$ is computed using $\mathbf{t}_t^l$ and $\mathbf{R}_t^l$ as shown in equation 2.27. Each pairing $\{\mathbf{x}_{lj}^{pix}(\tau), \mathbf{x}_{tj}^{pix}(\tau)\}$ is validated by the constraint equation

$$[\breve{\mathbf{x}}_{lj}^{pix}]^T(\tau)\mathbf{F}\breve{\mathbf{x}}_{tj}^{pix}(\tau) < \alpha \tag{3.3}$$

where $\breve{\mathbf{x}}_{lj}^{pix}$ denotes pixels in homogeneous coordinates and α is a threshold below which points are considered valid.

To determine matches across time, an initial set of matches is determined by matching the descriptors for the points extracted from the leading camera at times k and $k-1$. A homography is then computed with this initial set of matches in a RANSAC approach [10] to get rid of erroneous matches. Now each identified feature has image coordinates in four images. Figure 3.4 shows an example of the results of feature correspondence. It can be verified by eye that the matches corresponding to a single feature are consistent across cameras and across time epochs.

3.4.1 Image Masking.

An image mask is a binary matrix of the same size and shape as its corresponding image which indicates the regions in the image that should be processed for features. Image masks were used to mitigate erroneous feature matches in the feature correspondence algorithm and to prevent feature matches which would not be localized well by the triangulation algorithm. In a hallway, features tend to be densely packed on the outsides of the image: e.g. pictures, doors, fountains, etc. Features at the end of the hallway tend to be harder to localize. In addition, for the trail vehicle, part of its view does not intersect the lead vehicle's view.

Constructing the image mask for each image takes into account the estimated vanishing point for each image and the imaged vehicle target coordinates, in the case of the trail camera image. The end of the hallway can be estimated as a region surrounding

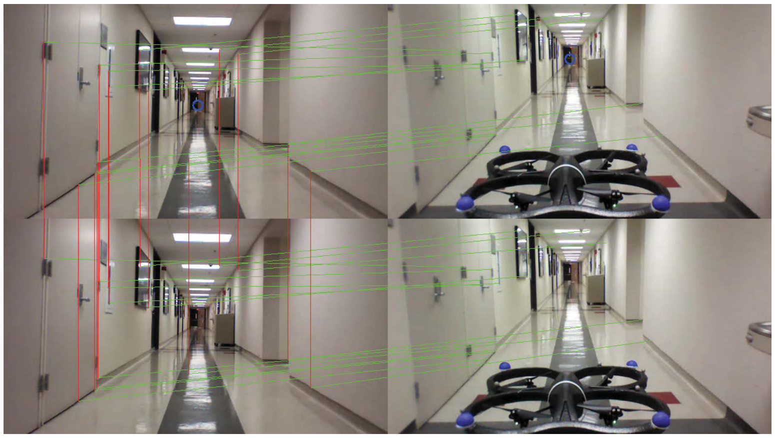


Figure 3.4: Results of Feature Correspondence. The image is a stitching of the images of the leading (left) and trailing (right) vehicles at the current time (top) and the previous time (bottom) epochs.Green lines indicate a match across cameras. Red lines indicate a match across time epochs.

the vanishing point in each image. The section of the trail camera image which is not in the lead camera's view can be estimated from the first and fourth vehicle target (refer to Figure 3.3). The image area below these targets is not in view of the lead vehicle. Figure 3.5 shows the result of applying the constructed image masks to the lead and trail camera images.

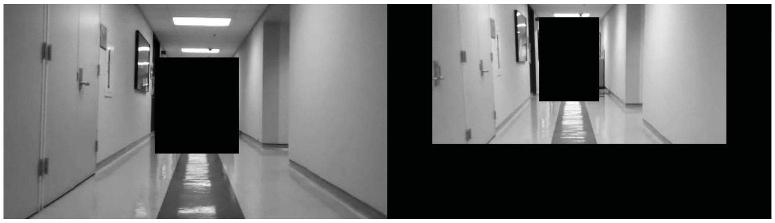


Figure 3.5: Image Processing Using Masks. After applying a mask to an image, the areas that are not of interest are black. Only the areas outside the mask will be processed for features to match

The benefit of using image masks is that it increases the probability of finding valid feature matches that will be well localized in the triangulation algorithm which follows (see Figure 3.2). The image masks ensure that the putative set of matches determined by the feature matching objects are more consistent with the overlap between the two images and eliminate some false matches before attempting to match across time epochs. Eliminating false matches during feature correspondence is preferred over eliminating false matches in later processes (triangulation and/or batch processing). It is possible to have a match which is not consistent with the image overlap, but satisfies the epipolar constraint and appears well localized. These types of matches are much more difficult to eliminate after the feature matching process, which makes computing the image masks an important step in feature correspondence.

3.5 Feature Triangulation

Feature triangulation uses each pair of lead and trail pixel coordinates contained in each vector $\mathbf{f}_j$, and the relative translation $\mathbf{t}_t^l$ and relative orientation DCM $\mathbf{R}_l^t$ (the transpose of $\mathbf{R}_t^l$) for each time epoch to determine the coordinates of each feature $\mathbf{M}_j$ in four frames of reference. These four frames of reference are the lead and trail camera frames for times k and $k-1$. While the coordinates of a feature referenced to the trail vehicle at time $k-1$ will be the only coordinates used in the next block, batch processing, triangulating features in all frames of references is another means of eliminating erroneous feature correspondences.

In the ideal case, any two matching image points would correspond to two camera frame pointing vectors which intersect at a distinct point in 3-space. However, due to errors in feature correspondence and relative camera pose estimation, this is hardly the case. Additionally, the feature correspondence algorithm detailed in section 3.4 will occasionally permit false matches.

3.5.1 Linear Triangulation .

While optimal algorithms exist for finding the most probable point in 3-space, the triangulation method developed in this section remains linear. The algorithm starts with two matched pointing vectors in the lead and trail camera frames $\mathbf{u}_l$, $\mathbf{u}_t$ which are found by applying the transform

$$\mathbf{u}_i = (\mathbf{T}_{c_i}^{pix})^T \check{\mathbf{x}}_{ij}^{pix} \tag{3.4}$$

where $\check{\mathbf{x}}_{ij}^{pix}$ is the pixel frame location of the j-th imaged feature from camera i in homogeneous coordinates. Equation 3.4 is essentially a transform from the pixel frame to the camera frame without applying depth. Let the epipolar plane $\mathbf{\Pi}$ be spanned by $\mathbf{u_t}$, $\mathbf{t}_t^l$ and take the trail camera frame as the frame of reference. The projection of $\mathbf{u_l}$ onto the epipolar plane is then [60]

$$\begin{aligned} \mathbf{\Pi} &= \begin{bmatrix} \mathbf{u}_t & \mathbf{t}_t^l \end{bmatrix} \\ \mathbf{u}_l' &= (\mathbf{\Pi}^T \mathbf{\Pi})^{-1} \mathbf{\Pi}^T \mathbf{R}_l^t \mathbf{u}_l \end{aligned} \tag{3.5}$$

Now the three vectors $\mathbf{u}_t$, $\mathbf{u}_l'$, $\mathbf{t}_t^l$ lie in a plane and satisfy the linear equation

$$s_t \cdot \mathbf{u}_t - s_l \cdot \mathbf{u}_l' = \mathbf{t}_t^l \tag{3.6}$$

where s_l and s_t are the estimated depths to the triangulated point in the lead and trail camera frames respectively. Equation 3.6 is an overdetermined system with one exact solution which is easily solved by QR decomposition [60]. Solutions which return negative can be discarded as they are not feasible (i.e. a feature cannot be behind the camera).

3.5.2 Measures of Accuracy.

False matches can be removed during triangulation by thresholding the projection error and parallax angle of the corresponding feature points. The projection error associated with triangulation is calculated by [60]

$$e = \|(\mathbf{I} - (\mathbf{\Pi}^T \mathbf{\Pi})^{-1} \mathbf{\Pi}^T) \mathbf{R}_l^t \mathbf{u}_l\| \tag{3.7}$$

which determines how much the pointing vectors needed to be adjusted to force their intersection. It was empirically determined that feature correspondences with more that 0.05 units projection error could be considered erroneous. Parallax angle refers to the angle between the two pointing vectors for a single feature. After $\mathbf{u}_t$ is projected onto the epipolar plane, the parallax angle can be estimated by

$$\theta = \arcsin(\|\mathbf{u}_t \times \mathbf{u}_l'\|) \tag{3.8}$$

Parallax angle will be further investigated in Chapter 4.

3.6 Batch Processing

The purpose of the batch processing algorithm is to create a single measurement from the outputs of feature correspondence, relative camera pose estimation, and feature triangulation, as shown in Figure 3.2, which can be fed directly into a Kalman filter. The batch processor designed in this section is an iterative non-linear model similar to the construction of the bundle adjustment problem in Section 2.10.4. The advantage of the batch processor over handling each input individually is that by incorporating all inputs at once, outliers can be handled before filtering. In addition, fewer measurements must be eventually processed by the Kalman filter. The obvious disadvantage is the processor's iterative solution.

A frame of reference for the batch processor must first be defined. Since the information given to the batch processor will be used to set up equations relating the lead and trail vehicle positions between times k and $k-1$, a local frame of reference must be chosen. The body frame of the trail vehicle at time $k-1$ will be the reference frame for the batch processor and will be denoted by a superscript w. The body frame for the other three vehicle positions in the measurement processing algorithm can be related to the measurement processor frame by the equations

$$\begin{aligned} \mathbf{C}_n^w &:= \mathbf{C}_n^{b_t(k-1)} \\ \mathbf{C}_{b_i(\tau)}^w &= \mathbf{C}_n^w \mathbf{C}_{b_i(\tau)}^n \end{aligned} \tag{3.9}$$

where $b_i(\tau)$ denotes the body frame of vehicle i at time τ. Each DCM on the right side of Equation 3.9 is computed using the estimated roll $\phi_i(\tau)$ and pitch $\theta_i(\tau)$ angles for each vehicle at time τ, which comes from each vehicle's navigation data stream, as well as the corrected yaw angle $\hat{\psi}_i(\tau)$ for each vehicle at time τ, which comes from the vanishing point algorithm.

Each quantity is related to the state variables by an equation of the form

$$\mathbf{z}(\tau) = \mathbf{h}(\mathbf{p}^w(\tau), \delta\boldsymbol{\Psi}(\tau), \mathbf{M}^w) + \mathbf{v}(\tau) \tag{3.10}$$

where $\mathbf{z}(\tau)$ is a realized measurement, $\mathbf{h}(\mathbf{p}^w(\tau), \delta\boldsymbol{\Psi}(\tau), \mathbf{M}^w)$ is a predicted measurement, $\mathbf{p}^w(\tau), \delta\boldsymbol{\Psi}(\tau), \mathbf{M}^w$ are the position(s), attitude error(s), and feature coordinates respectively and $\mathbf{v}(\tau)$ is zero-mean white Gaussian noise. The noise strength $\mathbf{R}$ is defined for each input as

$$\mathbf{R} = \mathrm{E}[\mathbf{v}(\tau)\mathbf{v}^T(\tau)] \tag{3.11}$$

where τ denotes time.

3.6.1 Feature Points .

The four image coordinates contained in each $\mathbf{f}_j$ (Section 3.4) are assumed independent and can be handled individually. Therefore, each feature point can be expressed as

$$\mathbf{x}_{ij}^{pix}(\tau) = \mathbf{h}_f(\mathbf{p}_i^w(\tau), \delta\mathbf{\Psi}_i(\tau), \mathbf{M}_j^w) + \mathbf{v}_f(\tau) \tag{3.12}$$

where $\mathbf{h}_f(\mathbf{p}_i^w(\tau), \delta\mathbf{\Psi}_i(\tau), \mathbf{M}_j^w)$, $\mathbf{x}_{ij}^{pix}(\tau)$ are the predicted and realized image coordinates of the j-th three-dimensional feature

$$\mathbf{M}_j^w = \begin{bmatrix} X_j^w & Y_j^w & Z_j^w \end{bmatrix}^T \tag{3.13}$$

for the i-th vehicle at time τ. The predicted image coordinates $\mathbf{h}_f(\mathbf{p}_i^w(\tau), \delta\mathbf{\Psi}_i(\tau), \mathbf{M}^w)$ can be calculated by the matrix equations [66]

$$\begin{aligned} \mathbf{s}_{ij}^{c_i}(\tau) &= \mathbf{C}_{b_i}^{c_i}\hat{\mathbf{C}}_w^{b_i(\tau)}(\mathbf{M}_j^w - \mathbf{p}_i^w(\tau)) \\ \mathbf{h}_f(\mathbf{p}_i^w(\tau), \delta\mathbf{\Psi}_i(\tau), \mathbf{M}_j^w) &= \frac{1}{[\mathbf{s}_{ij}^{c_i}(\tau)]_z}\mathbf{T}_{c_i}^{pix}\mathbf{s}_{ij}^{c_i}(\tau) \end{aligned} \tag{3.14}$$

where $\mathbf{s}_{ij}^{c_i}(\tau)$ is the pointing vector in the i-th vehicle's camera frame and $\mathbf{p}_i^w(\tau)$ is the position of the i-th vehicle at time τ in the measurement processor frame w. Note that the body to camera frame DCM for each vehicle $\mathbf{C}_{b_i}^{c_i}$ does not change over time, hence the absence of τ. The subscript z ($[\cdot]_z$) denotes the z component of the $\mathbf{s}_{ij}^{c_i}(\tau)$ vector. The corrected w-frame to body frame DCM $\hat{\mathbf{C}}_w^{b_i(\tau)}$ is computed using the attitude errors for vehicle i at time τ via the equation

$$\hat{\mathbf{C}}_w^{b_i(\tau)} = \mathbf{C}_w^{b_i(\tau)}[\mathbf{I} + \delta\mathbf{\Psi}_i(\tau)_\times] \tag{3.15}$$

where $\delta\boldsymbol{\Psi}_i(\tau)$ is a vector of the attitude errors for vehicle i at time τ. The influence matrices associated with $\mathbf{h}_f(\mathbf{p}_i^w(\tau), \delta\boldsymbol{\Psi}_i(\tau), \mathbf{M}_j^w)$ are found using chain rule derivatives as [66]

$$\mathbf{H}_{p_i f_{ij}}(\tau) = \frac{\partial \mathbf{h}_f(\mathbf{p}_i^w(\tau), \delta\boldsymbol{\Psi}_i(\tau), \mathbf{M}_j^w)}{\partial \mathbf{p}_i(\tau)} = \mathbf{T}_{c_i}^{pix} \cdot \frac{[\mathbf{s}_j^{c_i}(\tau)]_z \frac{\partial \mathbf{s}_j^{c_i}(\tau)}{\partial \mathbf{p}_i(\tau)} - \mathbf{s}_j^{c_i}(\tau)[\frac{\partial \mathbf{s}_j^{c_i}(\tau)}{\partial \mathbf{p}_i(\tau)}]_z}{[\mathbf{s}_j^{c_i}(\tau)]_z^2} \tag{3.16}$$

$$\frac{\partial \mathbf{s}_j^{c_i}(\tau)}{\partial \mathbf{p}_i(\tau)} = -\mathbf{C}_{b_i}^{c_i} \hat{\mathbf{C}}_w^{b_i(\tau)}$$

$$\mathbf{H}_{\Psi_i f_{ij}}(\tau) = \frac{\partial \mathbf{h}_f(\mathbf{p}_i^w(\tau), \delta\boldsymbol{\Psi}_i(\tau), \mathbf{M}_j^w)}{\partial (\delta\boldsymbol{\Psi}_i(\tau))} = \mathbf{T}_{c_i}^{pix} \cdot \frac{[\mathbf{s}_j^{c_i}(\tau)]_z \frac{\partial \mathbf{s}_j^{c_i}(\tau)}{\partial (\delta\boldsymbol{\Psi}_i(\tau))} - \mathbf{s}_j^{c_i}(\tau)[\frac{\partial \mathbf{s}_j^{c_i}(\tau)}{\partial (\delta\boldsymbol{\Psi}_i(\tau))}]_z}{[\mathbf{s}_j^{c_i}(\tau)]_z^2} \tag{3.17}$$

$$\frac{\partial \mathbf{s}_j^{c_i}(\tau)}{\partial (\delta\boldsymbol{\Psi}_i(\tau))} = -\mathbf{C}_{b_i}^{c_i} \mathbf{C}_w^{b_i(\tau)}[(\mathbf{M}_j^w - \mathbf{p}_i^w(\tau))]\times$$

$$\mathbf{H}_{M_j f_{ij}}(\tau) = \frac{\partial \mathbf{h}_f(\mathbf{p}_i^w(\tau), \delta\boldsymbol{\Psi}_i(\tau), \mathbf{M}_j^w)}{\partial \mathbf{M}_j^w} = \mathbf{T}_{c_i}^{pix} \cdot \frac{[\mathbf{s}_j^{c_i}(\tau)]_z \frac{\partial \mathbf{s}_j^{c_i}(\tau)}{\partial \mathbf{M}_j^w} - \mathbf{s}_j^{c_i}(\tau)[\frac{\partial \mathbf{s}_j^{c_i}(\tau)}{\partial \mathbf{M}_j^w}]_z}{[\mathbf{s}_{ij}^c]_z^2} \tag{3.18}$$

$$\frac{\partial \mathbf{s}_j^{c_i}(\tau)}{\partial \mathbf{M}_j^w} = \mathbf{C}_{b_i}^{c_i} \hat{\mathbf{C}}_w^{b_i(\tau)}$$

Once again, the subscript z ($[\cdot]_z$) denotes the z component of the vector. The error covariance for each feature point in units of pixels^2 is

$$\mathbf{R}_f = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \tag{3.19}$$

The error covariance values were chosen arbitrarily, but were found not to have a significant impact on solution accuracy if changed. This is because of the high accuracy of the relative camera position estimates, which are given a significantly higher weight in the batch processing algorithm such that the feature point error covariance becomes arbitrary.

3.6.2 Relative Camera Position .

The relative camera position estimates only take into account the relative translation between the lead and trail cameras at time τ. Therefore, each relative camera position can be expressed in meters as

$$\mathbf{t}_t^l(\tau) = \mathbf{h}_\Delta(\mathbf{p}_l^w(\tau), \mathbf{p}_t^w(\tau), \delta\mathbf{\Psi}_l(\tau), \delta\mathbf{\Psi}_t(\tau)) + \mathbf{v}_\Delta(\tau) \tag{3.20}$$

where $\mathbf{t}_t^l(\tau)$ comes from relative camera pose estimation (Section 3.3). The predicted relative camera position estimate $\mathbf{h}_\Delta(\mathbf{p}_l^w(\tau), \mathbf{p}_t^w(\tau), \delta\mathbf{\Psi}_l(\tau), \delta\mathbf{\Psi}_t(\tau))$ is calculated from the positions and attitude errors of both vehicles by the equation

$$\mathbf{h}_\Delta(\mathbf{p}_l^w(\tau), \mathbf{p}_t^w(\tau), \delta\mathbf{\Psi}_l(\tau), \delta\mathbf{\Psi}_t(\tau)) = \mathbf{C}_{b_t}^{c_t}\hat{\mathbf{C}}_w^{b_t(\tau)}(\mathbf{p}_l^w(\tau) - \mathbf{p}_t^w(\tau)) \tag{3.21}$$

where $\hat{\mathbf{C}}_w^{b_t(\tau)}$ was defined in Equation 3.15 and $\mathbf{C}_{b_t}^{c_t}$ will be described in Section 3.8.2. The vectors $\mathbf{p}_l^w(\tau)$ and $\mathbf{p}_t^w(\tau)$ represent the position of the lead and trail vehicles respectively at time τ expressed in the batch processor frame. This equation is nonlinear with respect to the attitude errors $\delta\mathbf{\Psi}_l(\tau)$, $\delta\mathbf{\Psi}_t(\tau)$. The influence matrices associated with $\mathbf{h}_\Delta(\mathbf{p}_l^w(\tau), \mathbf{p}_t^w(\tau), \delta\mathbf{\Psi}_l(\tau), \delta\mathbf{\Psi}_t(\tau))$ are found via chain rule as

$$\mathbf{H}_{p_l\Delta}(\tau) = \frac{\partial\mathbf{h}_\Delta(\mathbf{p}_l^w(\tau), \mathbf{p}_t^w(\tau), \delta\mathbf{\Psi}_l(\tau), \delta\mathbf{\Psi}_t(\tau))}{\partial\mathbf{p_l}(\tau)} = \mathbf{C}_{b_t}^{c_t}\hat{\mathbf{C}}_w^{b_t(\tau)} \tag{3.22}$$

$$\mathbf{H}_{p_t\Delta}(\tau) = \frac{\partial\mathbf{h}_\Delta(\mathbf{p}_l^w(\tau), \mathbf{p}_t^w(\tau), \delta\mathbf{\Psi}_l(\tau), \delta\mathbf{\Psi}_t(\tau))}{\partial\mathbf{p_t}(\tau)} = -\mathbf{C}_{b_t}^{c_t}\hat{\mathbf{C}}_w^{b_t(\tau)} \tag{3.23}$$

$$\mathbf{H}_{\Psi_t\Delta}(\tau) = \frac{\partial\mathbf{h}_\Delta(\mathbf{p}_l^w(\tau), \mathbf{p}_t^w(\tau), \delta\mathbf{\Psi}_l(\tau), \delta\mathbf{\Psi}_t(\tau))}{\partial(\delta\mathbf{\Psi}_t(\tau))} = \mathbf{C}_{b_t}^{c_t}\mathbf{C}_w^{b_t(\tau)}[\mathbf{p}_l^w(\tau) - \mathbf{p}_t^w(\tau)]_\times \tag{3.24}$$

$$\mathbf{H}_{\Psi_l\Delta}(\tau) = 0 \tag{3.25}$$

The error covariance for each relative camera position estimate $\mathbf{R}_\Delta$ will be characterized in the next chapter.

3.6.3 Batch Solver.

The state of the batch estimator contains the position and attitude errors for the lead and trail vehicle at the current time k; the position and attitude errors of the lead vehicle at the previous time $k-1$; and the features such that the state can be written as

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}_l^w(k) & \delta\boldsymbol{\Psi}_l(k) & \mathbf{p}_t^w(k) & \delta\boldsymbol{\Psi}_t(k) & \mathbf{p}_l^w(k-1) & \delta\boldsymbol{\Psi}_l(k-1) & \mathbf{M}_1^w & \dots & \mathbf{M}_j^w \end{bmatrix} \tag{3.26}$$

The body frame of the trail vehicle at time $k - 1$ is the reference frame for the state of the batch estimator and the position of the trail vehicle at time $k - 1$ is the origin. The batch estimator seeks a solution vector $\mathbf{x}$ which maximizes the distribution [28]

$$p(\mathbf{x}|\mathbf{z}) = (2\pi)^{-\frac{n}{2}}\boldsymbol{\Sigma}_{\mathbf{z}}^{-\frac{1}{2}}\exp\left(-\frac{1}{2}[\mathbf{f}(\mathbf{x}) - \mathbf{z}]^T\boldsymbol{\Sigma}_{\mathbf{z}}^{-1}[\mathbf{f}(\mathbf{x}) - \mathbf{z}]\right) \tag{3.27}$$

where $\mathbf{z}$ is a vector of relative camera positions and feature points

$$\mathbf{z} = \begin{bmatrix} \mathbf{t}_t^l(k) \\ \mathbf{t}_t^l(k-1) \\ \mathbf{x}_{l1}^{pix}(k) \\ \vdots \\ \mathbf{x}_{tj}^{pix}(k-1) \end{bmatrix} \tag{3.28}$$

$\mathbf{f}(\mathbf{x})$ is a vector of the predicted relative camera positions and feature points

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \mathbf{h}_{\Delta}(\mathbf{p}_l^w(k), \mathbf{p}_t^w(k), \delta\boldsymbol{\Psi}_l(k), \delta\boldsymbol{\Psi}_t(k)) \\ \mathbf{h}_{\Delta}(\mathbf{p}_l^w(k-1), \mathbf{p}_t^w(k-1), \delta\boldsymbol{\Psi}_l(k-1), \delta\boldsymbol{\Psi}_t(k-1)) \\ \mathbf{h}_f(\mathbf{p}_l^w(\tau), \delta\boldsymbol{\Psi}_l(\tau), \mathbf{M}_1^w) \\ \vdots \\ \mathbf{h}_f(\mathbf{p}_t^w(\tau), \delta\boldsymbol{\Psi}_t(\tau), \mathbf{M}_j^w) \end{bmatrix} \tag{3.29}$$

and $\boldsymbol{\Sigma}_{\mathbf{z}}$ is a Gaussian covariance representing the error between $\mathbf{z}$ and $\mathbf{f}(\mathbf{x})$ and is formed by

$$\mathbf{\Sigma_z} = \begin{bmatrix} \mathbf{R}_\Delta & & & & \\ & \mathbf{R}_\Delta & & & \\ & & \mathbf{R}_f & & \\ & & & \ddots & \\ & & & & \mathbf{R}_f \end{bmatrix} \tag{3.30}$$

The solution vector can be expressed as a perturbation $\Delta\mathbf{x}$ about a nominal state $\mathbf{x}_0$

$$\mathbf{x} = \mathbf{x}_0 + \Delta\mathbf{x} \tag{3.31}$$

Using equation 3.31, the nonlinear function $\mathbf{f}(\mathbf{x})$ can be expanded using the Taylor series to form

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}_0) + \mathbf{J}\Delta\mathbf{x} \tag{3.32}$$

where $\mathbf{J}$ is the Jacobian of $\mathbf{f}(\mathbf{x})$ evaluated at the nominal state. The Jacobian is constructed in pieces using the influence matrices outlined in Sections 3.6.1 and 3.6.2. As an example, for two relative camera position estimates (for time k and $k-1$) and a single feature, the Jacobian has the form

$$\mathbf{J} = \begin{bmatrix} \mathbf{H}_{p_l\Delta}(k) & \mathbf{0} & \mathbf{H}_{p_t\Delta}(k) & \mathbf{H}_{\Psi_t\Delta}(k) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_{p_l\Delta}(k-1) & \mathbf{0} & \mathbf{0} \\ \mathbf{H}_{p_l f_{l1}}(k) & \mathbf{H}_{\Psi_l f_{l1}}(k) & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_{M_1 f_{l1}}(k) \\ \mathbf{0} & \mathbf{0} & \mathbf{H}_{p_t f_{t1}}(k) & \mathbf{H}_{\Psi_t f_{t1}}(k) & \mathbf{0} & \mathbf{0} & \mathbf{H}_{M_1 f_{t1}}(k) \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_{p_l f_{l1}}(k-1) & \mathbf{H}_{\Psi_l f_{l1}}(k-1) & \mathbf{H}_{M_1 f_{l1}}(k-1) \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_{M_1 f_{t1}}(k-1) \end{bmatrix} \tag{3.33}$$

Using Least Squares techniques, equation 3.32 can be rearranged to form the normal equations [60]

$$(\mathbf{J}^T \mathbf{\Sigma}_{\mathbf{z}}^{-1} \mathbf{J}) \Delta \mathbf{x} = \mathbf{J}^T \mathbf{\Sigma}_{\mathbf{z}}^{-1} (\mathbf{z} - \mathbf{f}(\mathbf{x})) \tag{3.34}$$

and the system of equations can be solved to find $\Delta\mathbf{x}$. At each iteration, the resulting $\Delta\mathbf{x}$ is added to $\mathbf{x}$ until the vector converges, at which point the distribution in Equation 3.27 is maximized.

3.6.3.1 Levenburg-Marquardt Solution.

The Levenburg-Marquardt method introduces an augmentation to the normal equations, so that they become [28]

$$\left(\mathbf{J}^T \mathbf{\Sigma}_{\mathbf{z}}^{-1} \mathbf{J} + \lambda \cdot \mathrm{diag}(\mathbf{J}^T \mathbf{\Sigma}_{\mathbf{z}}^{-1} \mathbf{J})\right) \Delta \mathbf{x} = \mathbf{J}^T \mathbf{\Sigma}_{\mathbf{z}}^{-1} (\mathbf{z} - \mathbf{f}(\mathbf{x})) \tag{3.35}$$

and the system is solved in same way as before. The key to the Levenburg-Marquardt method is in the update of the λ parameter. Although there are many argued best methods for updating this parameter, the method chosen in this work is simple, and follows the method outlined in [28]. If $\Delta\mathbf{x}$ increases with iteration, increase λ by a factor of 10 and recompute $\Delta\mathbf{x}$ without recomputing the Jacobian $\mathbf{J}$ and prediction $\mathbf{f}(\mathbf{x})$. If $\Delta\mathbf{x}$ decreases with iteration, reduce λ by a factor of 10 and continue.

3.6.3.2 Robust Cost Functions.

Built into the Levenburg-Marquardt method is the minimization of the squared norm of the residuals

$$\delta_i = \mathbf{z}_i - \mathbf{f}_i(\mathbf{x}) \tag{3.36}$$

where δ_i is the residual for the *i*-th measurement $\mathbf{z}_i$. Minimizing the squared norm is sufficient when the data is outlier-free. However, the feature correspondence algorithm developed in Section 3.4 and triangulation algorithm developed in Section 3.5 will occasionally forward large outliers to the batch processor. It is therefore necessary to weight the residual vector using a cost function which is more robust to outliers. The cost function used in this thesis is the Huber cost function, which de-weights the cost of outlier residuals according to the equation [28]

$$C(\delta_i) = \begin{cases} \delta_i^2 & : |\delta_i| < b \\ 2b|\delta_i| - b^2 & : \text{otherwise} \end{cases} \tag{3.37}$$

where b is approximately the outlier threshold.

The cost function can be integrated into the Levenbur-Marquardt algorithm by weighting the residuals for each measurement by [28]

$$\begin{aligned} w_i &= \frac{C(||\delta_i||)^{(1/2)}}{||\delta_i||} \\ \delta_i' &= w_i \delta_i \end{aligned} \tag{3.38}$$

and the new residual vector is formed by concatenating all of the δ_i''s.

3.6.4 Kalman Filter Measurement Formation .

The result of the batch processor at this point are a nominal state **x** and the Jacobian **J** evaluated at the nominal state. The first order approximation for the covariance of the nominal state is simply [28]

$$\mathbf{P}_{xx} = (\mathbf{J}^T \mathbf{\Sigma}_{\mathbf{z}}^{-1} \mathbf{J})^\dagger \tag{3.39}$$

where † denotes the pseudo-inverse in the case that $\mathbf{J}^T \mathbf{\Sigma}_{\mathbf{z}}^{-1} \mathbf{J}$ is not invertible.

The measurement forwarded to the Kalman filter is determined by a linear transformation of the nominal state **x**, excluding the feature states

$$\mathbf{z}_{BATCH} = \begin{bmatrix} \mathbf{p}_l^w(k) \\ \mathbf{p}_t^w(k) \\ \mathbf{p}_l^w(k-1) \end{bmatrix} = \mathbf{T}\mathbf{x} \tag{3.40}$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{I} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{I} & 0 \end{bmatrix} \tag{3.41}$$

Similarly, the covariance of the state $\mathbf{P}_{xx}$, excluding all terms corresponding to the feature states, can be transformed by the equation

$$\mathbf{R}_{BATCH} = \mathbf{T}\mathbf{P}_{xx}\mathbf{T}^T \tag{3.42}$$

and the measurement can now be fed into the Kalman filter.

3.7 Navigation Kalman Filter

The navigation Kalman filter developed in this work fuses measurements from the batch processor and vanishing point algorithm with a linear system model in order to produce an estimate of the position of both vehicles in the navigation frame. The roll and pitch angles $\phi_i(\tau)$, $\theta_i(\tau)$, and the corrected yaw angle $\hat{\psi}_i(\tau)$ are used to rotate the measurement from the batch processing algorithm into the navigation frame.

3.7.1 System Model.

The system model for the two vehicle network is a linear stochastic differential equation of the form

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{G}(t)\mathbf{w}(t) \tag{3.43}$$

where $\mathbf{x}(t)$ is the system state, $\mathbf{w}(t)$ is zero-mean white Gaussian noise, and $\mathbf{F}(t)$, $\mathbf{G}(t)$ are the state dynamics and noise matrices which characterize the relationships between the system state and noise processes. As mentioned in [18], the ARDrone's internal stabilization algorithm makes capturing the vehicle's dynamics difficult and thus an appropriate approximation is a constant velocity model.

The state of a single vehicle i is represented by the vector

$$\mathbf{x}_i(t) = \begin{bmatrix} X_i^n & Y_i^n & Z_i^n & V_{x_i}^n & V_{y_i}^n & V_{z_i}^n & \delta\psi_i & \delta\dot{\psi}_i \end{bmatrix}^T \tag{3.44}$$

where (X_i^n, Y_i^n, Z_i^n) is the position of the i-th vehicle in the navigation frame, $(V_{x_i}^n, V_{y_i}^n, V_{z_i}^n)$ is the velocity of the i-th vehicle in the navigation frame, and $\delta\psi_i$, $\delta\dot{\psi}_i$ are the heading error and heading error rate for the i-th vehicle. The pitch and roll errors, $\delta\theta$, $\delta\phi$, are not modelled as they are not observable from the measurements. The state dynamics matrix for the i-th vehicle

$$\mathbf{F}_i(t) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.45}$$

characterizes the constant velocity nature of the system model by setting the derivatives of the velocity states to zero.

Noise is allowed to propagate through the system by entering the velocity, heading error, and heading error rate states. The noise processes are assumed to be independent of each other, leading to the noise vector for the i-th vehicle

$$\mathbf{w}_i(t) = \begin{bmatrix} w_{V_x} & w_{V_y} & w_{V_z} & w_{\delta\psi} & w_{\delta\dot{\psi}} \end{bmatrix}^T \tag{3.46}$$

It follows that the noise matrix is then

$$\mathbf{G}_i(t) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.47}$$

Assuming that the states and noise processes for each vehicle are also independent from one another, the complete system model for the two vehicle network is

$$\overset{\cdot}{\begin{bmatrix} \mathbf{x}_l(t) \\ \mathbf{x}_t(t) \end{bmatrix}} = \begin{bmatrix} \mathbf{F}_l(t) & \mathbf{0} \\ \mathbf{0} & \mathbf{F}_t(t) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x}_l(t) \\ \mathbf{x}_t(t) \end{bmatrix} + \begin{bmatrix} \mathbf{G}_l(t) & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_t(t) \end{bmatrix} \cdot \begin{bmatrix} \mathbf{w}_l(t) \\ \mathbf{w}_t(t) \end{bmatrix} \tag{3.48}$$

where the subscripts l and t indicate variables for the lead and trail vehicles as defined in Equations 3.44 - 3.47. The noise strength for each vehicle is defined by

$$\mathbf{Q}_i(t) = \begin{bmatrix} 0.0625 & 0 & 0 & 0 & 0 \\ 0 & 0.0625 & 0 & 0 & 0 \\ 0 & 0 & 0.0625 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0.01 \end{bmatrix} \tag{3.49}$$

which also indicates that the noise variables are assumed independent from each other. The non-zero elements of $\mathbf{Q}_i(t)$ were chosen as a starting point for tuning the filter and will be investigated in the next chapter. They were selected because they were the initial values of the noise strength matrix in [18].

3.7.2 *Vanishing Points.*

Vanishing point measurements are a scalar which directly measure the heading error state at time t_k. The measurement can be simply expressed as

$$\delta\psi_i^{MEAS}(t_k) = h_{\delta\psi}(\mathbf{x}_i(t_k)) + v_{\delta\psi}(t_k) \tag{3.50}$$

$$h_{\delta\psi}(\mathbf{x}_i) = \delta\psi_i(t_k) \tag{3.51}$$

where $\delta\psi_i^{MEAS}(t_k)$ is the heading error reported by the vanishing point estimation algorithm and $\delta\psi_i$ is simply the heading error state of the i-th vehicle, as shown in Equation 3.44. The influence matrix associated with $h_{\delta\psi}(\mathbf{x}_i(t_k))$ is simply

$$\mathbf{H}_{\delta\psi_i} = \frac{\partial h_{\delta\psi}(\mathbf{x}_i)}{\partial(\delta\psi_i(t_k))} = 1 \tag{3.52}$$

The measurement error covariance for each vanishing point measurement is

$$\mathbf{R}_{\delta\psi} = 0.04 \tag{3.53}$$

in units of rad^2. The covariance value was chosen to be the same value as used in [18].

3.7.3 *Delayed State Extended Kalman Filter Update.*

Using the delayed state algorithm described in section 2.7.2, the measurement described in Section 3.6.4 and the vanishing point measurements described in the previous section can be integrated into the Kalman filter. First, the nonlinear measurement equation is

$$\mathbf{z} = \mathbf{h}(\mathbf{x}(t_k), \mathbf{x}(t_{k-1})) + \mathbf{v} \tag{3.54}$$

where $\mathbf{z}$ is a vector of measurements

$$\mathbf{z} = \begin{bmatrix} \delta\psi_l^{MEAS}(t_k) & \delta\psi_t^{MEAS}(t_k) & \mathbf{z}_{BATCH}^T \end{bmatrix}^T \tag{3.55}$$

and

$$\mathbf{h}(\mathbf{x}(t_k), \mathbf{x}(t_{k-1})) = \begin{bmatrix} \delta\psi_l(t_k) \\ \delta\psi_t(t_k) \\ \mathbf{C}_{\tilde{n}}^{b_t(t_{k-1})}\mathbf{C}_n^{\tilde{n}}(\mathbf{p}_l^n(t_k) - \mathbf{p}_t^n(t_{k-1})) \\ \mathbf{C}_{\tilde{n}}^{b_t(t_{k-1})}\mathbf{C}_n^{\tilde{n}}(\mathbf{p}_t^n(t_k) - \mathbf{p}_t^n(t_{k-1})) \\ \mathbf{C}_{\tilde{n}}^{b_t(t_{k-1})}\mathbf{C}_n^{\tilde{n}}(\mathbf{p}_l^n(t_{k-1}) - \mathbf{p}_t^n(t_{k-1})) \end{bmatrix} \tag{3.56}$$

where $\mathbf{p}_l^n(t_k)$, $\mathbf{p}_l^n(t_{k-1})$,$\mathbf{p}_t^n(t_k)$, $\mathbf{p}_t^n(t_{k-1})$ are the positions at time t_k and t_{k-1} respectively and $\mathbf{C}_{\tilde{n}}^{b_t(t_{k-1})}$ is the transpose of the body to nav DCM computed by Equation 2.4 using the body angles $\phi_t(t_{k-1}), \theta_t(t_{k-1}), \hat{\psi}_t(t_{k-1})$. The DCM $\mathbf{C}_n^{\tilde{n}}$ is a function of the trail vehicle heading error at time t_{k-1} and can be calculated from

$$\mathbf{C}_n^{\tilde{n}} = \begin{bmatrix} \cos(\delta\psi_t(t_{k-1})) & -\sin(\delta\psi_t(t_{k-1})) & 0 \\ \sin(\delta\psi_t(t_{k-1})) & \cos(\delta\psi_t(t_{k-1})) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.57}$$

Equation 3.56 can then be linearized to form the influence matrices

$$\mathbf{H}_k = \begin{bmatrix} \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 1 & 0 & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 0 & 0 \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 0 & 0 & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 1 & 0 \\ \mathbf{C}_{\tilde{n}}^{b_t(t_{k-1})}\mathbf{C}_n^{\tilde{n}} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{C}_{\tilde{n}}^{b_t(t_{k-1})}\mathbf{C}_n^{\tilde{n}} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} \end{bmatrix} \tag{3.58}$$

$$\mathbf{H}_{k-1} = \begin{bmatrix} \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 0 & 0 & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 0 & 0 \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 0 & 0 & \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & 0 & 0 \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & -\mathbf{C}_{\tilde{n}}^{b_t(t_{k-1})}\mathbf{C}_n^{\tilde{n}} & \mathbf{0}_{3\times3} & \mathbf{A}_1 & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & -\mathbf{C}_{\tilde{n}}^{b_t(t_{k-1})}\mathbf{C}_n^{\tilde{n}} & \mathbf{0}_{3\times3} & \mathbf{A}_2 & \mathbf{0}_{3\times1} \\ \mathbf{C}_{\tilde{n}}^{b_t(t_{k-1})}\mathbf{C}_n^{\tilde{n}} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times1} & -\mathbf{C}_{\tilde{n}}^{b_t(t_{k-1})}\mathbf{C}_n^{\tilde{n}} & \mathbf{0}_{3\times3} & \mathbf{A}_3 & \mathbf{0}_{3\times1} \end{bmatrix} \tag{3.59}$$

$$
\begin{aligned}
\mathbf{A}_1 &= \mathbf{C}_{\tilde{n}}^{b_t(t_{k-1})} \frac{\partial}{\partial(\delta\psi)} \mathbf{C}_n^{\tilde{n}} (\mathbf{p}_t(t_k) - \mathbf{p}_t(t_{k-1})) \\
\mathbf{A}_2 &= \mathbf{C}_{\tilde{n}}^{b_t(t_{k-1})} \frac{\partial}{\partial(\delta\psi)} \mathbf{C}_n^{\tilde{n}} (\mathbf{p}_t(t_k) - \mathbf{p}_t(t_{k-1})) \\
\mathbf{A}_3 &= \mathbf{C}_{\tilde{n}}^{b_t(t_{k-1})} \frac{\partial}{\partial(\delta\psi)} \mathbf{C}_n^{\tilde{n}} (\mathbf{p}_t(t_{k-1}) - \mathbf{p}_t(t_{k-1}))
\end{aligned}
$$

where $\frac{\partial}{\partial(\delta\psi)} \mathbf{C}_n^{\tilde{n}}$ can be calculated as

$$
\frac{\partial}{\partial(\delta\psi)} \mathbf{C}_n^{\tilde{n}} = \begin{bmatrix} -\sin(\delta\psi_t(t_{k-1}) & -\cos(\delta\psi_t(t_{k-1}) & 0 \\ \cos(\delta\psi_t(t_{k-1}) & -\sin(\delta\psi_t(t_{k-1}) & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.60}
$$

The measurement covariance for the delayed state update is nominally

$$
\mathbf{R} = \begin{bmatrix} \mathbf{R}_{\delta\psi} & 0 & 0 \\ 0 & \mathbf{R}_{\delta\psi} & 0 \\ 0 & 0 & \mathbf{R}_{BATCH} \end{bmatrix} \tag{3.61}
$$

in the case where the vanishing point estimation algorithm returns two vanishing point measurements and the batch processing algorithm is able to produce a single measurement.

3.8 Procedures

This section describes a number of necessary calibration steps. Where possible, these procedures were automated in the ROS framework.

3.8.1 Camera Calibration.

Camera calibration was conducted prior to each flight using a publicly available ROS camera calibration package. The camera calibration package uses OpenCV camera calibration code, which is based on the algorithms found in [9] and [69]. The camera calibration package subscribes to a topic containing the image stream of the camera to be calibrated. The image stream is processed using an OpenCV function which automatically detects a chessboard pattern and estimates the inner corners. Figure 3.6 is a screen capture of the ROS package capturing a chessboard in an image. The camera calibration package can determine when there is enough image data to constrain the calibration equations and return valid calibration parameters.

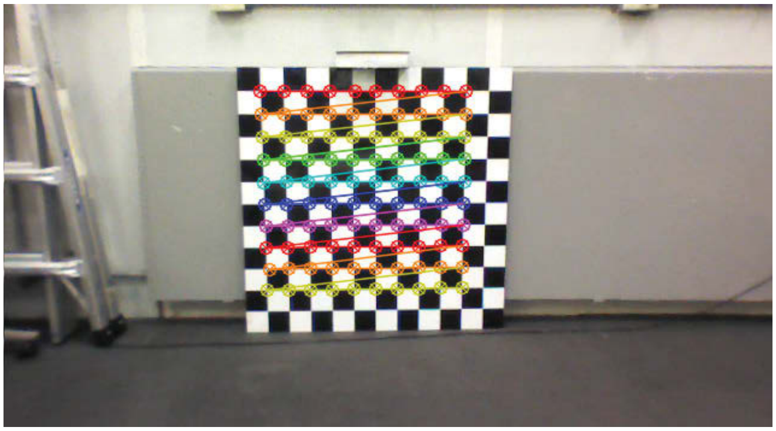


Figure 3.6: Automatic Camera Calibration. The ROS camera calibration package automates camera calibration by recognizing a calibration pattern.

3.8.2 Tilt Calibration .

In order to keep the lead vehicle in its field of view, a tilt angle was added to the trail camera. The tilt needs to be calibrated to improve the accuracy of the image processing and filtering algorithms. Like camera calibration, tilt calibration uses the calibration board shown in Figure 3.6. The chessboard auto-detection will detect the same 10x10 grid of points – shown as colored circles – in each image. The grid points belong to an object coordinate frame where the *x*-axis runs left to right, the *y*-axis runs top to bottom, and the *z*-axis points towards the wall. The object coordinates of each grid point are then

$$\mathbf{p} = \begin{bmatrix} x_c - l \cdot (i-1) & y_c - l \cdot (j-1) & 0 \end{bmatrix}^T \tag{3.62}$$

where i is the column number, j is the row number, l is the length of each square in the chessboard pattern, and (x_c, y_c) is the center of the grid.

Using the camera matrix and distortion coefficients found from camera calibration, the grid point coordinates, and the image location of each grid point, OpenCV's `solvePnP` function can be used to the find the relative orientation between the camera frame and the calibration board frame. By placing the trail vehicle on a level surface and ensuring the calibration board is perpendicular to the floor and roughly centered in the image, the relative orientations correspond to the tilt angles of the trail camera. These angles are computed for each image in the calibration sequence and averaged.

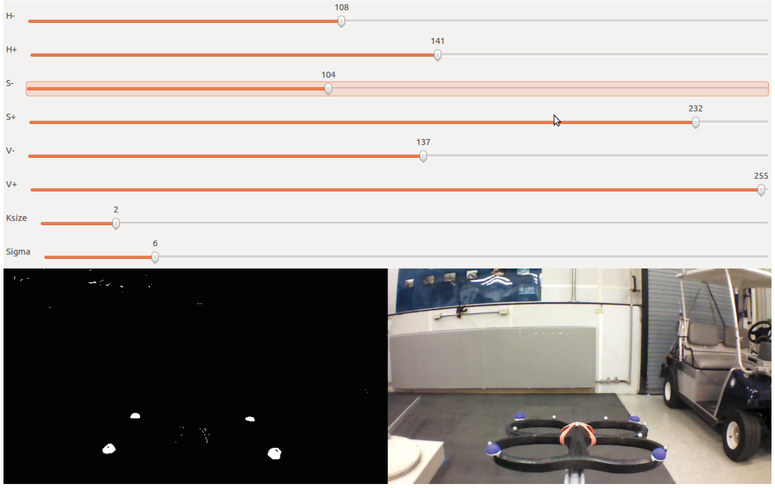
The rotations about the y and z axes were found to be negligible; roughly 0° for both axes. However, the rotation α about the x axis, which is the axis about which the camera was rotated, is of interest. After calibration of α the DCM relating the body frame and camera frame of the trail vehicle $\mathbf{C}_{c_t}^{b_t}$ can be calculated by the equation

$$\mathbf{C}_{c_t}^{b_t} = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \mathbf{C}_c^b \tag{3.63}$$

where $\mathbf{C}_c^b$ is equivalent to the body to camera DCM described in section 2.4. The calibrated tilt angle α was found to be $-11.5°$. The DCM for relating the body and camera frame of the lead vehicle $\mathbf{C}_{c_l}^{b_l}$ is simply $\mathbf{C}_c^b$.

3.8.3 Targeting Calibration.

Although the backlit targets are less sensitive to changes in lighting conditions during a data collection, the color ranges of the targets on the lead vehicle need to be calibrated before each collection to ensure that the targets can be reliably tracked. This is especially important when operating in different environments (e.g the Vicon chamber v. the hallway). Figure 3.7 shows the target calibration GUI designed to calculate the necessary HSV ranges and blur parameters for target tracking.



H- 108
H+ 141
S- 104
S+ 232
V- 137
V+ 255
Ksize 2
Sigma 6

Figure 3.7: Calibration of the Target Detector. The GUI designed allows the user to manipulate the HSV ranges and blur parameters using OpenCV trackbars. The parameters should be adjusted so that targets are clearly distinguishable in the binary image (bottom left).

It is not necessary to calibrate the target detector to the point where all non-target regions are completely blocked out. Instead, the values should be adjusted so that a balance between target area and background noise is found. The parameters are saved in a YAML file and loaded into the detector at run time.

3.8.4 Data Collection.

Data is captured using a ROS tool called a bag. ROS bags store serialized data from messages as they are received [2]. The bag can be played back to the same topics they were recorded from. In addition, the bag can be played at a slower rate, paused, and configured from a launch file. The utility of ROS bags allows for seamless transitioning between online capture and post-processing of data.

3.9 Summary

This chapter developed the software, algorithms, and tools used in this thesis. The first section outlined the system architecture. The next sections developed the target tracking algorithm, the relative camera pose estimation algorithm, the feature correspondence algorithm, and the feature triangulation algorithm which yield estimates of relative camera positions and feature locations. A Levenburg-Marquardt batch processing algorithm was also developed to process relative pose and feature correspondence measurements into a single estimate of the motion of both vehicles over one time epoch. The delayed state Kalman filter equations were then developed to estimate vehicle position based on vanishing point and batch processor measurements. The final section described procedures for calibration prior to running experiments. The next chapter will present a set of experiments which validate the operation of the system and its algorithms described in this chapter.

IV. Results

In chapter 3, a system of algorithms were developed to enable cooperative localization for two airborne vehicles. This chapter presents a set of experiments used to validate the algorithms developed in this thesis. The experiments progress in stages, proceeding from evaluating the performance of the individual algorithms, up to evaluating the performance of the entire cooperative binocular stereopsis algorithm.

4.1 Testing Equipment

When running experiments the two ARDrone vehicles were mounted on a rigid cart assembly, shown in Figure 4.1. The cart ensures that the targets on the lead vehicle are within view of the trail vehicle's camera at all times. The wireless router also rests on the cart assembly, so that both vehicles are always within a short range of the router.

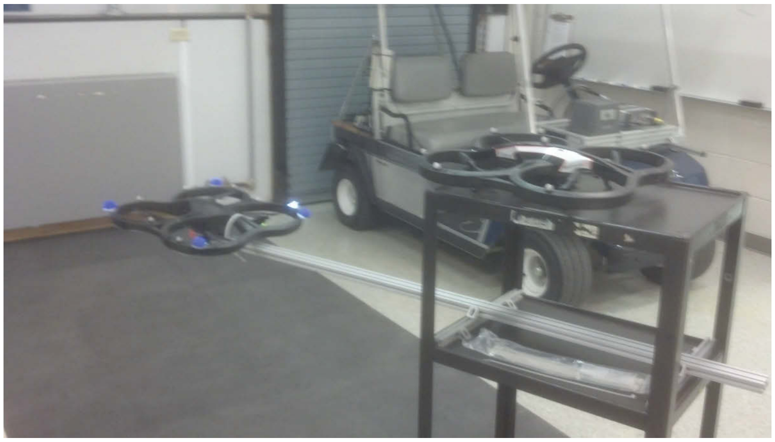


Figure 4.1: Testing Equipment. The equipment shown was used to keep the lead vehicle with illuminated targets in view of the trailing vehicle during tests.

It is important to note that although the vehicles were mounted, this information was not given to any of the algorithms. The developed algorithms re-estimate the relative position and attitude of the two vehicles for each image pair as if their relative positions were changing with time.

4.2 Feature Matching Experiment

A number of different algorithms for feature matching are provided within OpenCV. Specifically, OpenCV provides objects for feature extraction, feature descriptor extraction, and feature descriptor matching. Each object has its own set of tuning parameters which can be configured in their construction or adjusted during run time [10]. It is unrealistic to attempt a comparison of feature extractors, descriptors, and matchers – there are arguably many combinations which could achieve good performance with the right tuning. Instead, an extractor, descriptor, and matcher were chosen and tuned until an acceptable performance was reached.

The two vehicles were mounted on the cart assembly and their image feeds were recorded to two synchronized video files. The cart was allowed to move so that the test of the feature matching algorithm was more robust. The video files were then processed to find feature matches for each image pair in the video files. For this test, matching was performed at the descriptor level – the distance between feature descriptors is the only criteria for determining a match.

4.2.1 Feature Matching Experiment Results.

For this research, the FAST [64] feature extractor, BRISK [40] descriptor extractor, and Brute Force Hamming matcher [10] were used for feature matching. These feature and descriptor extractors were chosen because of their low computational cost and comparable performance to SIFT and SURF. To overcome the fact that FAST is not inherently scale-invariant, a Gaussian pyramid adapter [10] was used to modify the feature extraction algorithm to allow for scale-space feature extraction. Table 4.1 shows the values of each tuning parameter for the extractor, descriptor, and matcher selected.

Table 4.1: Matching Parameters.

FAST	Threshold	15
	Normax Suppression	false
	Pyramid Levels	3
BRISK	Threshold	1
	Octaves	3
	Pattern Scale	2.2
Brute Force Matcher	Cross-check	true

Table 4.2 shows the matching performance achieved for a sample image sequence after tuning the selected feature matching objects and Gaussian pyramid adapter. The results indicate that on average OpenCV's feature matching objects will return enough putative feature correspondences to continue with the feature correspondence algorithm.

Table 4.2: Matching Performance.

Frames Processed	103
Avg Detection Time (ms)	1.242
Avg Extraction Time (ms)	2.525
Avg Matching Time (ms)	1.533
Avg Features Per Frame	207
Avg Matches Per Frame	96

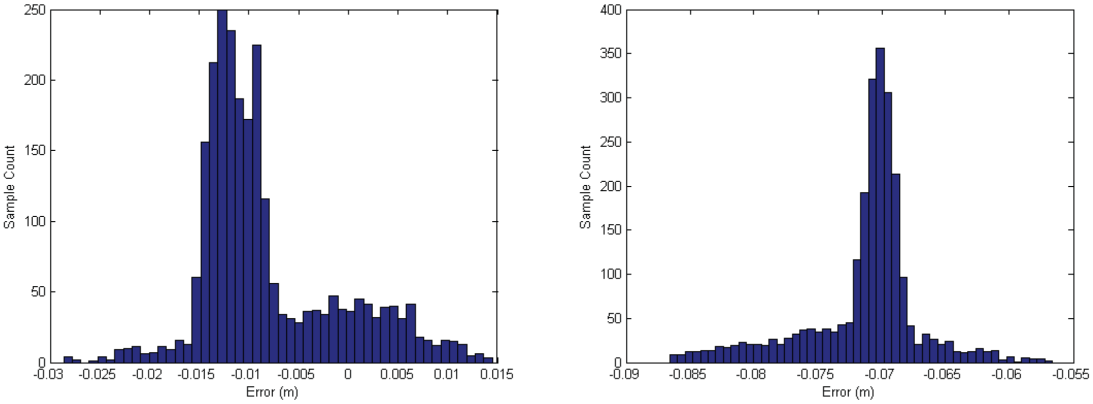
4.3 Relative Camera Pose Estimation Accuracy Experiment

The goal of this experiment was to characterize the error associated with estimating the six variables defining relative camera pose (three for translation and three for orientation). In addition, the results this test will be used to construct the error covariance of the relative camera position estimate $\mathbf{R}_{\Delta}$. The vehicles were initially mounted on the cart assembly in the Vicon chamber. Next, the target areas were initialized so that the target tracking algorithm could lock on to the vehicle targets on the lead vehicle. Once a lock was acquired, the trail vehicle could be moved around; however, movement needed to be gradual to prevent the target tracking algorithm from losing lock. A ROS synchronizer was used to match up relative pose estimates with Vicon motion capture data and write the information to a CSV file. Finally, the data file was processed in MATLAB to determine the error in the relative camera pose estimation algorithm.

4.3.1 Relative Camera Pose Estimation Accuracy Experiment Results.

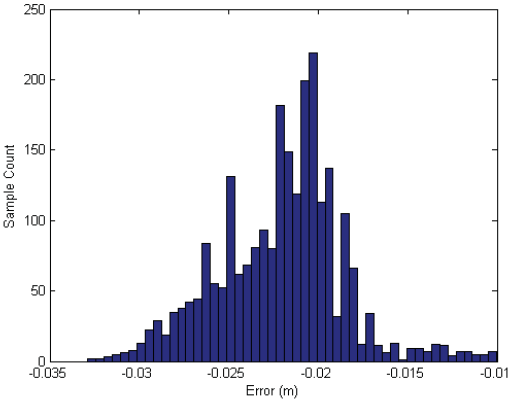
Figures 4.2 and 4.3 show the distributions for 2500 samples of the error between the translation and rotation estimated by the image processing algorithm and the true translation and rotation calculated from Vicon motion capture data.

The bias in the errors are likely due to errors in the trail camera model and the measured target locations. The target locations were measured by hand and are likely only accurate within a few centimeters. The distributions for the translation estimation errors do not appear to be Gaussian; however, the distributions are almost symmetrical about their means and their spreads are relatively small such that Gaussian approximations can be applied. The X error distribution has a mean of -0.008 m and a standard deviation of 0.007 m. The Y error is has a mean of -0.071 m and a standard deviation of 0.005 m. The Z error distribution has a mean of -0.022 m and a standard deviation of 0.004 m. The measurement error covariance for relative camera position estimates is therefore



(a) X Error

(b) Y Error



(c) Z Error

Figure 4.2: Distributions of Relative Translation Errors.

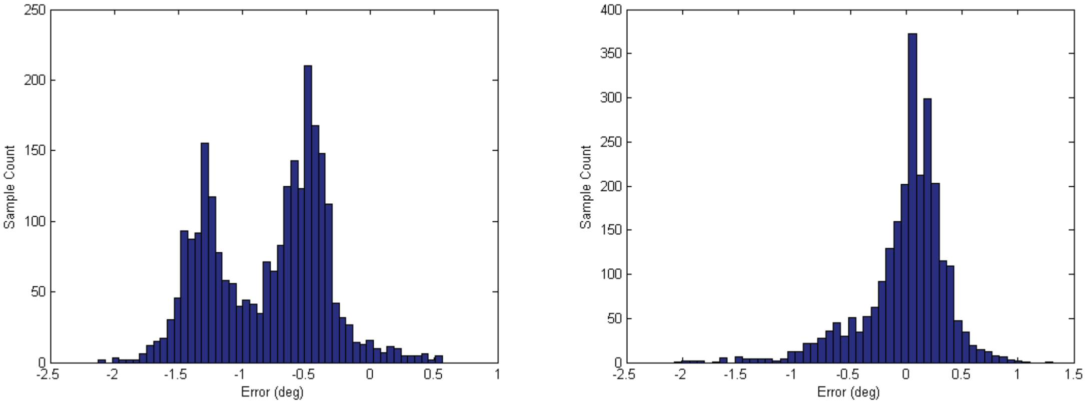
$$\mathbf{R}_\Delta = \begin{bmatrix} 0.000049 & 0 & 0 \\ 0 & 0.000025 & 0 \\ 0 & 0 & 0.000016 \end{bmatrix} \tag{4.1}$$

and the mean errors can be subtracted from the relative translation estimate to yield an unbiased estimate.

Similar to the biases in the translation estimates, there are small biases in the rotational estimates. These biases are also likely due to camera calibration errors and errors in the measured target locations. The distributions of the rotation estimation errors appear less well-behaved than the translational errors. However, their spreads are also relatively small such that the means can be used to calibrate the rotation estimate. The X-axis rotation error has a mean of -0.796° and a standard deviation of 0.459°. The Y-axis rotation error has a mean of -0.004° and a standard deviation of 0.380°. The Z-axis rotation error has a mean of -0.336° and a standard deviation of 0.861°. Although the relative orientation is not used as a measurement, it is used with the translation estimate to calculate the fundamental matrix between paired images. The estimated relative orientation and translation are also used to triangulate features, as described in Section 3.5.1.

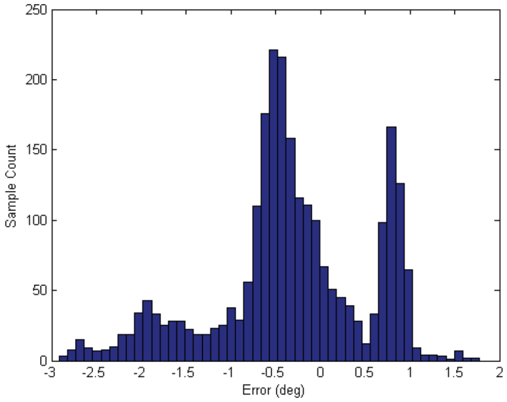
4.4 Triangulation Accuracy Experiment

The goal of this experiment was to characterize the error in the triangulation algorithm presented in Section 3.5.1. Features in the hallway were hand selected from images, as shown in Figure 4.4. The baseline and relative orientation for the two cameras were estimated using OpenCV's `solvePnP` function, using the `CV_ITERATIVE` option.



(a) X-Axis Rotation Error

(b) Y-Axis Rotation Error



(c) Z-Axis Rotation Error

Figure 4.3: Distributions of Relative Orientation Errors.

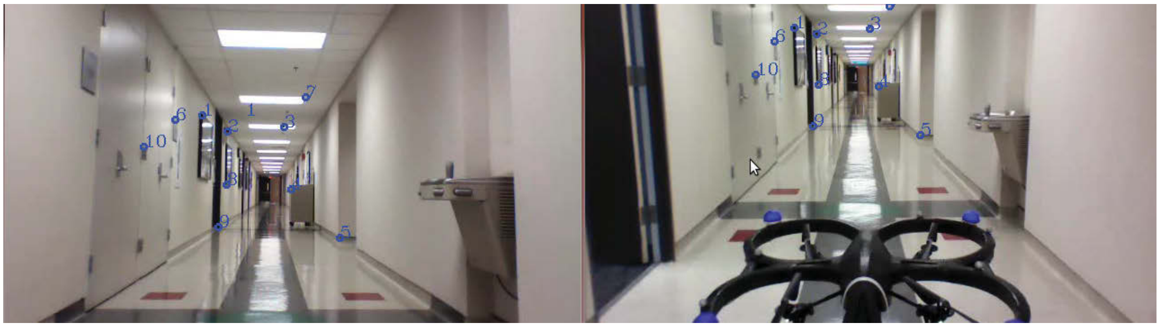


Figure 4.4: Sample Points for Evaluating Triangulation Accuracy. Corresponding points were hand selected from two images.

Since points were sampled in the hallway, there was no access to truth data from Vicon. Instead, a laser range-finder was used to determine the true depth to each target feature. The depth to each sampled feature is the least observable parameter in triangulation, and consequently has the most error. The dominating factor for determining the depth of a feature is the parallax angle between both cameras' line of sight vectors. Figure 4.5 illustrates the relationship between parallax angle and depth estimation error.

Clearly Figure 4.5 shows that there is a high correlation between parallax angle and depth estimation error. Features correspondences with a parallax angle less than 0.06 radians may have up to 3 m of depth error. This lower bound on the parallax angle was used in the image processing algorithm to determine if a three dimensional feature could be accurately triangulated from its pair of image measurements.

One unanticipated product of this test was the indication of an upper bound on the parallax for real feature correspondences. From Figure 4.5, the maximum achievable parallax for the image pair is nearly 0.18 radians. This limit on the parallax angle can be used to discard erroneous matches which appear to have an extremely good parallax angle. Figure 4.6 shows an example of such a match.

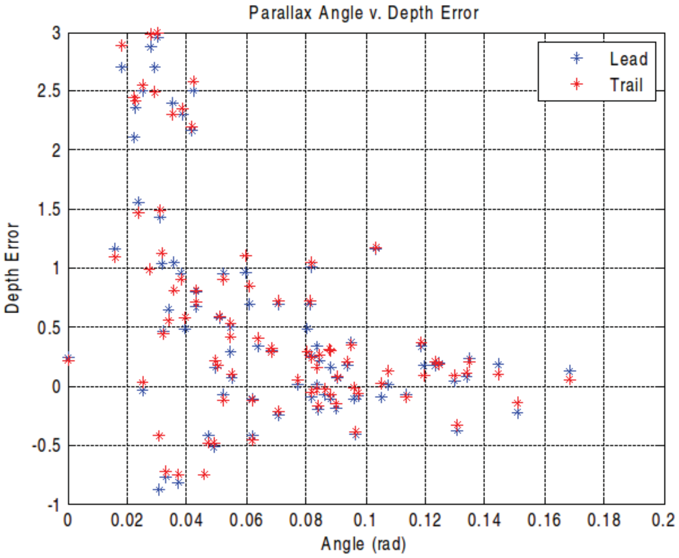


Figure 4.5: Feature Depth Error v. Parallax Angle. The error in depth determined by the triangulation algorithm and by a laser range finder are plotted against the parallax angle.

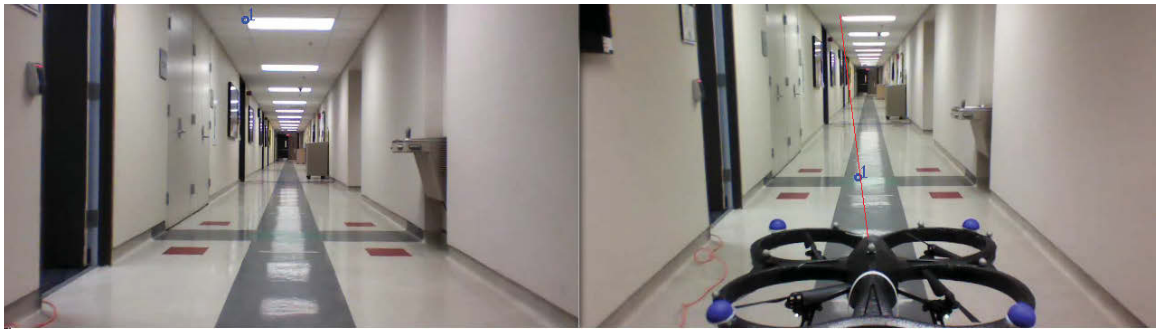


Figure 4.6: An Erroneous Match With Good Parallax Angle. This figure shows a possible scenario in which the intensity of a light fixture is erroneously matched to refelctions on a floor tile. The image coordinates of the light fixture in the left image map to a line which passes through the true feature location and converges at the epipole in the second image.

Typically, the epipolar constraint can be used to determine erroneous matches. However, the epipolar constraint maps a point in one image to a line of possible corresponding points in another image. The mistmatched points shown in Figure 4.6 pass the epipolar constraint and their computed parallax angle is 0.4530 radians. Enforcing a maximum limit on parallax angle ensures that matches like the one shown in Figure 4.6 are discarded.

4.5 Batch Processing Experiment

The goal of the this test was to determine the accuracy of the batch processor in the hallway environment. To do this, the two ARDrone vehicles were mounted on the cart assembly and pushed down the hallway at a constant velocity and heading. Figure 4.7 illustrates the desired course for this test. An important distinction between the testing in this section and the testing done in [18] is that the test course did not need to be modified prior to testing. The feature correspondence algorithm is able to capture enough natural features in the environment to extract a positional measurement.

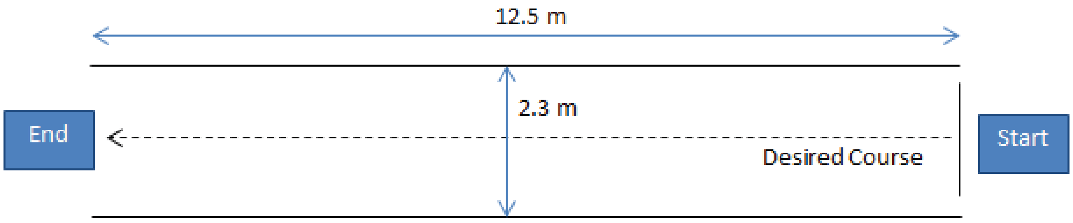


Figure 4.7: Hallway Course Used for Testing the Batch Processing Algorithm.

Because the vehicles were mounted to the cart assembly and the heading is nominally aligned with the center of the hallway, it was assumed that motion would only be observable in the $+X$ direction. The average velocity in the X-direction was computed by dividing the course length by the time. The average change in position between frames in the X direction, ΔX_{avg}, could then be approximated by the equation

$$\Delta X_{avg} = \frac{average\ velocity}{frame\ rate} \tag{4.2}$$

and the average change in position between frames for the other two directions was assumed to be zero. In addition, each data file used in this test was clipped to a region of frames in which the motion approximation would hold.

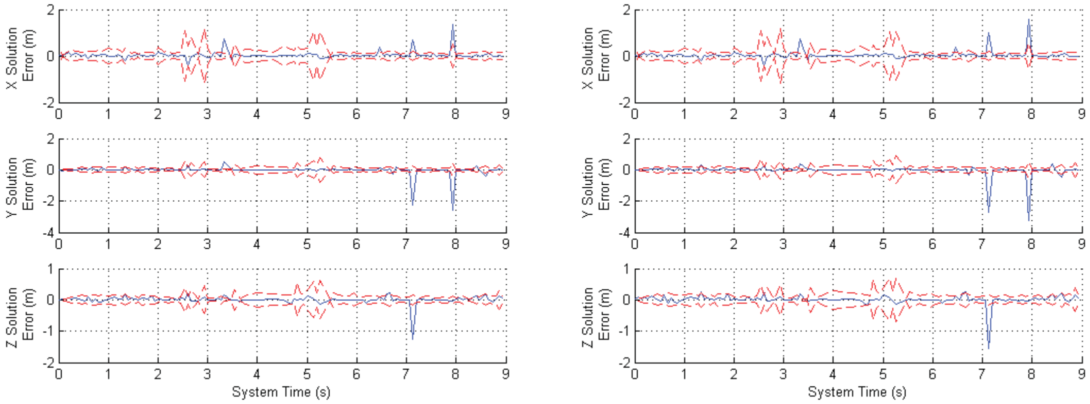
4.5.1 Batch Processing Experiment Results.

The hallway test was carried out for four trials. The results presented in this section are from a single trial, but are representative of the performance of the batch processor in each trial. Figure 4.8 shows the solution errors over time, found by subtracting the average motion per frame from the solution returned by the batch processor.

Notice that the solution errors in Figure 4.8c are much lower than the errors shown in Figures 4.8a and 4.8b. Since the body frame of the trail vehicle at time t_{k-1} is the refrence frame for the batch processor, the relative camera position estimate which relates the position of the lead vehicle to the position of the trail vehicle at time t_{k-1} is an absolute measurement of the position of the lead vehicle at time t_{k-1} in this reference frame. It is therefore expected, and indeed verified, that the solution errors in Figure 4.8c should be on the order of the level of accuracy of relative camera position estimate.

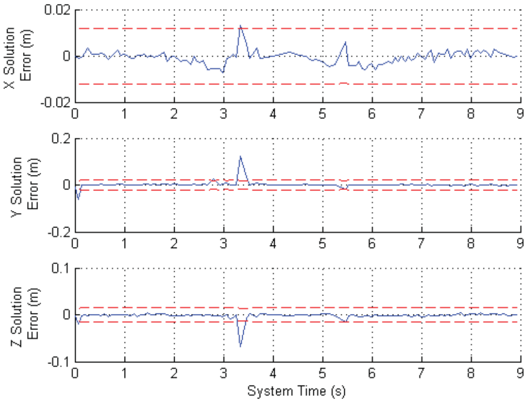
The solution errors appear to be zero mean and are well bounded by 3-σ standard deviation as computed by the batch processor, with exception to a few easily identifiable outliers. This indicates that the batch processor is able to accurately characterize the precision in its solution. It was found, however, that solutions in which the covariance deviates too far from the average cause the Kalman filter's estimate to be thrown off. Each of the (X, Y, Z) triplets in Figure 4.8 can be handled separately from eachother with the following constraint

$$\sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2} < \beta \tag{4.3}$$



(a) Solution Error $\mathbf{p}_l^w(k)$

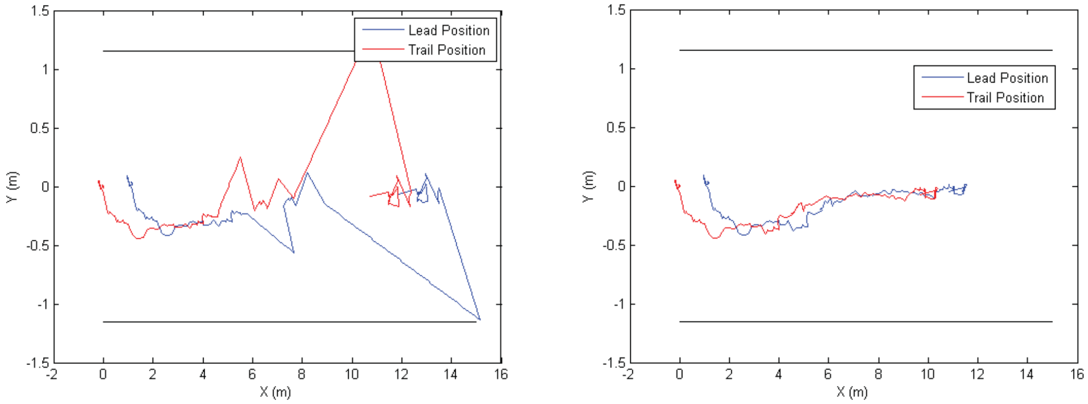
(b) Solution Error $\mathbf{p}_t^w(k)$



(c) Solution Error $\mathbf{p}_l^w(k-1)$

Figure 4.8: Batch Processor Solution Errors. The batch processor solution errors (blue) are plotted with 3-σ standard deviation lines (red).

which is essentially the magnitude of the three-dimensional position uncertainty for each position in the batch solution state. The value of β was empirically chosen to be 0.2 for the first triplet, 0.2 for the second triplet, and 0.01 for the third triplet. Solutions which exceed any of the these constraints are discarded. Figure 4.9 shows the improvement in the Kalman filter estimate after applying these constraints. In both instances where the filter estimate begins to diverge, the solution errors exceed one of the empirical thresholds.



(a) Filter Estimate Without Constraints

(b) Filter Estimate With Constraints

Figure 4.9: Effect of Enforcing Solution Error Constraints. The X-Y trajectories estimated by the Kalman filter with (left) and without (right) constraints on the batch solution error.

It it more pertinent to analyze the statistical properties of the measurement which is given to the Kalman filter. Figure 4.10 shows the distribution of measurement errors. Since the covariance of the measurement is recomputed for each iteration, the measurement error is divided by its standard deviation to normalize the errors to the same distribution. The histogram in Figure 4.10 closely follows a standard zero-mean Gaussain distribution. This is important, since the delayed state Kalman filter assumes that the error in the measurements follows a Gaussian distribution.

4.6 Kalman Filter Experiments

The goal of these experiments was to determine the accuracy of the delayed state Kalman filtering algorithm in phases. First, the Kalman filter will be validated independently from the batch processor. Then the Kalman filter will be validated including the batch processor. Due to an implementation issue, the Kalman filtering algorithm was duplicated in MATLAB. There appeared to be numerical issues in

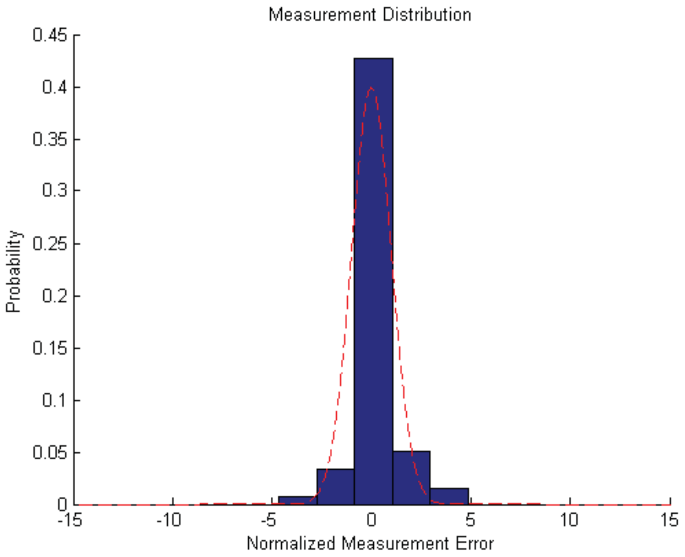


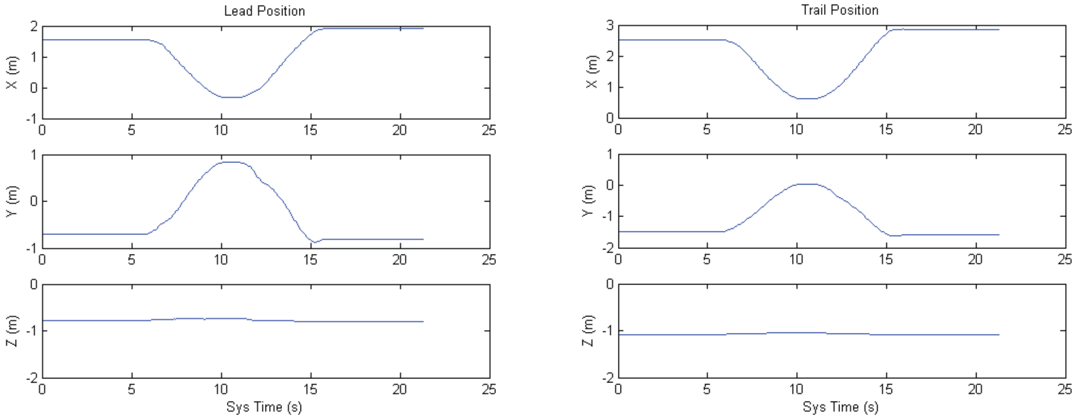
Figure 4.10: Normailized Measurement Error Distribution. This figure shows a histogram of measurement errors (blue) normalized by their standard deviation and a trace of a standard zero-mean Gaussian distribution (red).

computing the Kalman gain using OpenCV's linear algebra which caused the filter to reject too many measurements. In each experiment, data is first processed in C++ for measurements, then imported into MATLAB to be used by the Kalman filter algorithm.

4.6.1 Monte Carlo Simulations.

The purpose of this test was to isolate the Kalman filtering algorithm from the batch processing algorithm and determine the filter's accuracy with generated measurements. The flight profile in Figure 4.11 was captured by the Vicon system and has milimeter level accuracy.

Simulated measurements were formed by transforming the truth data into a sequence of exact (error-free) measurements and then adding Gaussian noise. The Gaussian noise was generated from a distribution with constant covariance – not a changing covariance as would be expected if the filter were connected to the batch processor. The covariance was



(a) Lead Flight Profile

(b) Trail Flight Profile

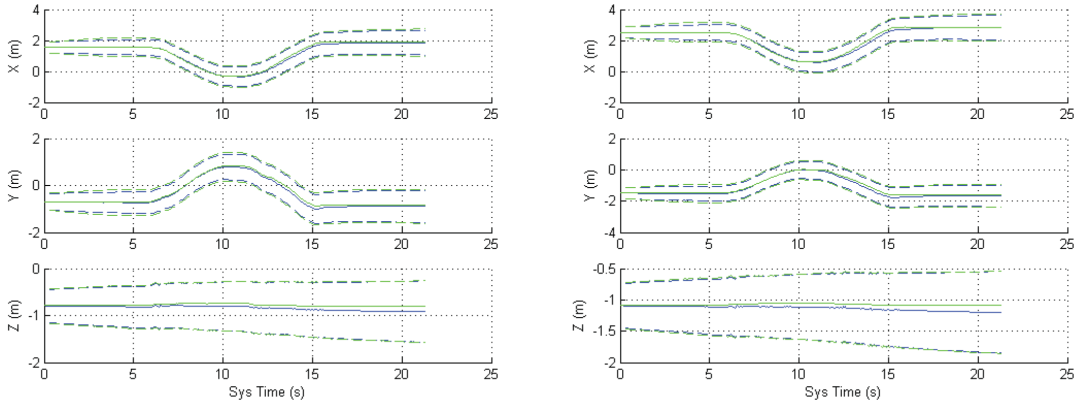
Figure 4.11: Flight Profile for Monte Carlo Simulations. The flight profile exhibits forward and backward motion in two directions as well as periods of no motion.

set to a reasonable estimate of the average covariance computed by the batch processor. Because the covariance does not vary like the real data, adequate performance in this test is a necessary, but not sufficient condition to validate the Kalman filtering algorithm.

4.6.2 Monte Carlo Simulation Results.

The Monte Carlo simulation consisted of 1000 individual trials. The initial state for each trail was also given random Gaussian error. Figure 4.12 shows the ensemble mean of the filter computed trajectories, the true trajectory, the filter computed standard deviation, and the ensemble standard deviation of the filter computed trajectories.

It is clear from these results that on average the Kalman filter can be expected to closely follow the true trajectory, given that the measurements do not deviate much from the measurement covariance used to generate measurements in this experiment. The Monte Carlo simulation also reveals a primary weakness of the filtering algorithm. Over time, the uncertainty in the filter's estimate grows, most noticeably in the Z direction. This is because measurements correspond to differences between two states of the filter –



(a) Lead Estimate

(b) Trail Estimate

Figure 4.12: Monte Carlo Kalman Filter Simulation Results. Estimates for the position of the lead (a) and trail (b) vehicles. The ensemble mean trajectory (solid blue) and filter computed standard deviation (dashed blue) are plotted against the true trajectory (solid green) and ensemble standard deviation (dashed green)

contributing only relative information. Because absolute information is not brought into the filter, the filter can only reduce the rate of error growth. Overcoming this issue is not within the scope of this research and will be left for future work.

4.6.3 Processor Integration Test Setup.

The purpose of this test was to investigate the performance of the Kalman filtering algorithm by using outputs from the batch processor and the corrected body angles. The two ARDrone vehicles were mounted on the cart assembly and placed at the starting position of the test course shown in Figure 4.13. For each sample run, the cart assembly was pushed at a constant rate and heading down the center of the test course and the image and navigation data was saved to a ROS bag file.

To process the bag file, the image and navigation data were converted into two synchronized videos and a CSV file with time tagged navigation measurements (i.e. the

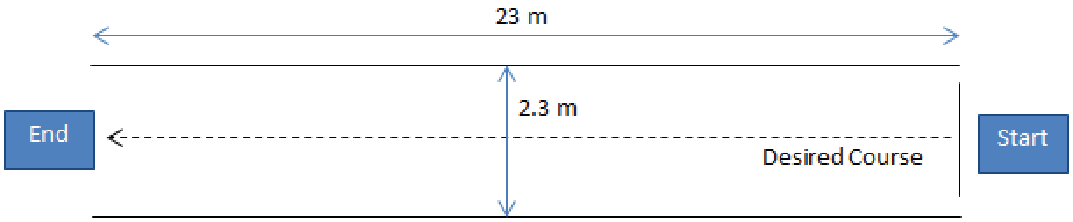
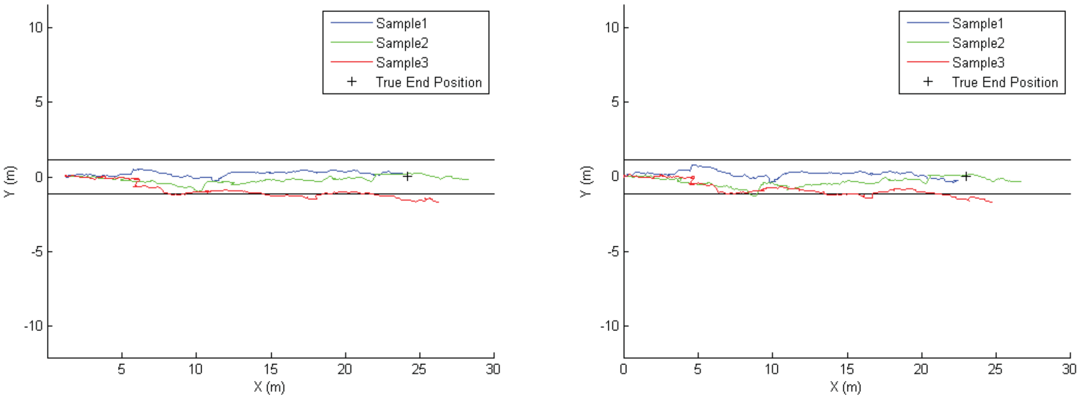


Figure 4.13: Hallway Course Used for Testing the Integration of the batch processor and Kalman Filter.

reported body angles). This ensured that the software would always process the same information. Next, the image sequences and navigation measurements were processed for measurements and corrected body angles. For each image pair, the vanishing points were extracted and used to correct the error in the reported yaw angle. The images were also processed for matching features and the batch processor was used to extract the batch measurement. The measurement and body angle data was then uploaded into MATLAB and processed in the Kalman filtering algorithm to produce an estimate of the vehicle trajectories. The test course was sampled three times, each at a desired rate of 15 Hz. For each sample, the first relative pose estimate was used to initialize the Kalman filter state.

4.6.4 Unfiltered Trajectory Estimate.

As a point of comparison, Figure 4.14 first presents the estimated X-Y trajectories for each sample based on taking the cumulative sum of the delta-position information contained in the measurements (i.e. without filtering the measurements). As an intermediate step, outliers were identified and removed from the measurements. Without position information from another source, it is not possible to determine the error in the trajectory estimate over time. Instead, Table 4.3 summarizes the final RMS position errors.



(a) Lead Vehicle Trajectory

(b) Trail Vehicle Trajectory

Figure 4.14: Trajectories Estimated From Measurements Only. The trajectories calculated by accumulating raw measurement data are plotted for the lead and trail vehicles. The solid black lines indicate the walls of the hallway.

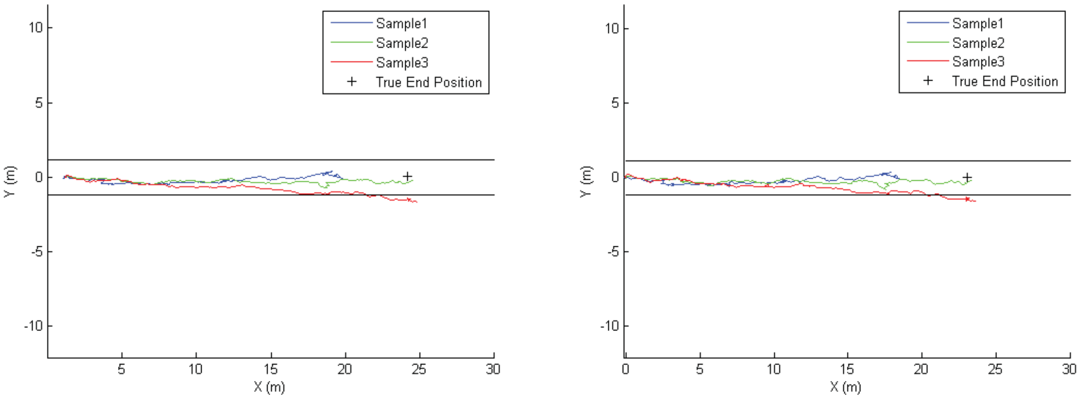
Table 4.3: Final Position Errors Using Measurements Only.

Sample	Lead RMS Error (m)	Trail RMS Error (m)
1	2.807	3.372
2	4.412	4.072
3	3.429	3.907

Figure 4.14 shows that the vehicle motion captured by the measurements is consistent with what would be expected. In sample three, however, there appears to be a persistent bias in the *Y* direction that causes the raw trajectories to drift outside the hallway limits. The errors in the final position, as indicated in Table 4.3, are likely due to the accumulation of noise as the delta-position information is summed. The Kalman filter is designed to handle noise in the measurements and should produce a better estimate of the trajectories.

4.6.5 Processor Integration Results .

Figure 4.15 shows the *X*-*Y* trajectories computed by the Kalman filter for each sample. The trajectories computed by the filter agree with the motion captured by the measurements. In addition, each trajectory computed by the filter is noticeably less noisy than the trajectory estimated from the cumulative sum of raw measurement data. This is because the Kalman gain limits the amount of information that each measurement contributes to the trajectory estimate. The filter, however, is unable to deal with the measurement drift in sample three.

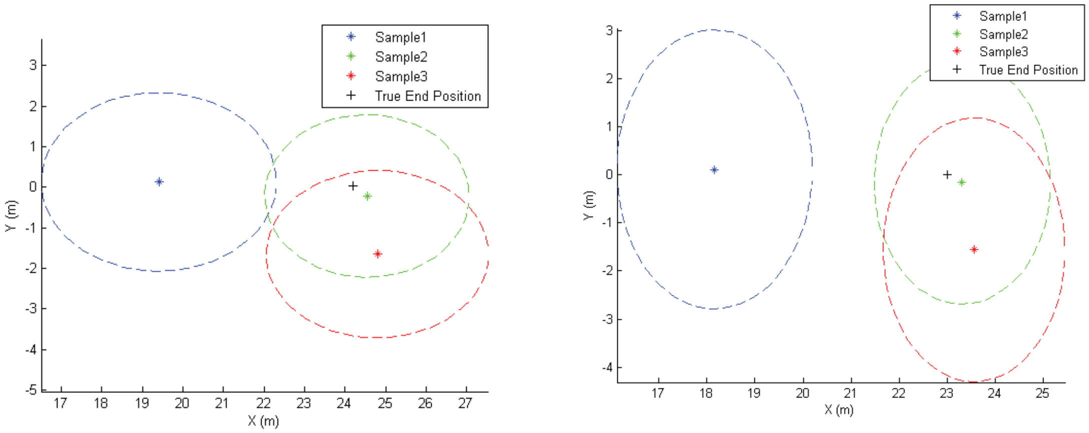


(a) Lead Vehicle Trajectory

(b) Trail Vehicle Trajectory

Figure 4.15: Kalman Filter Position Estimate. The filter computed trajectories are plotted for the lead and trail vehicles. The solid black lines indicate the walls of the hallway.

Figure 4.16 compares the measured final position of each vehicle to the filter's estimate, and the RMS errors are summarized in Table 4.4. Figure 4.16 and Table 4.4 show that there is an inconsistency between the true error and the error estimated by the filter for sample one, while the solutions are consistent for the other two samples. This could be due to several factors, which will be investigated in the rest of this section.



(a) Final Position Estimate (Lead)

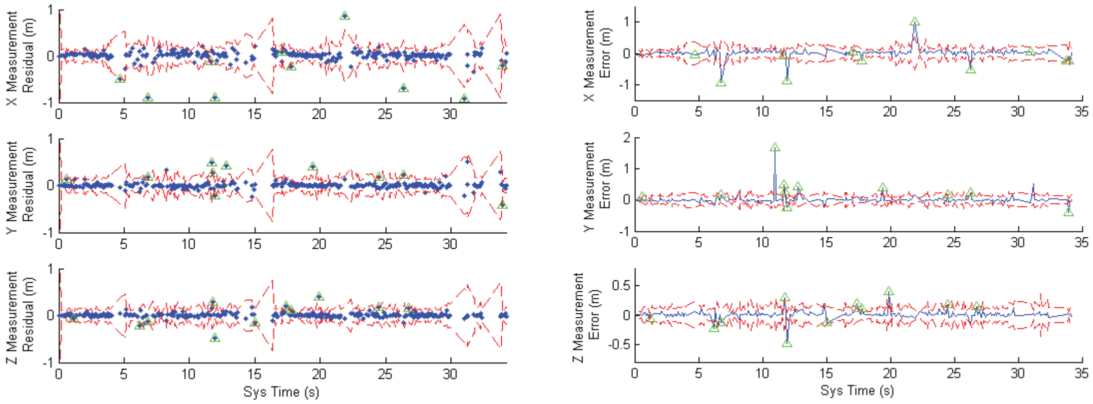
(b) Final Position Estimate (Trail)

Figure 4.16: Final Position Estimates. The final position estimate for each sample (asterisks) are plotted with 3-σ computed X-Y standard deviation ellipses.

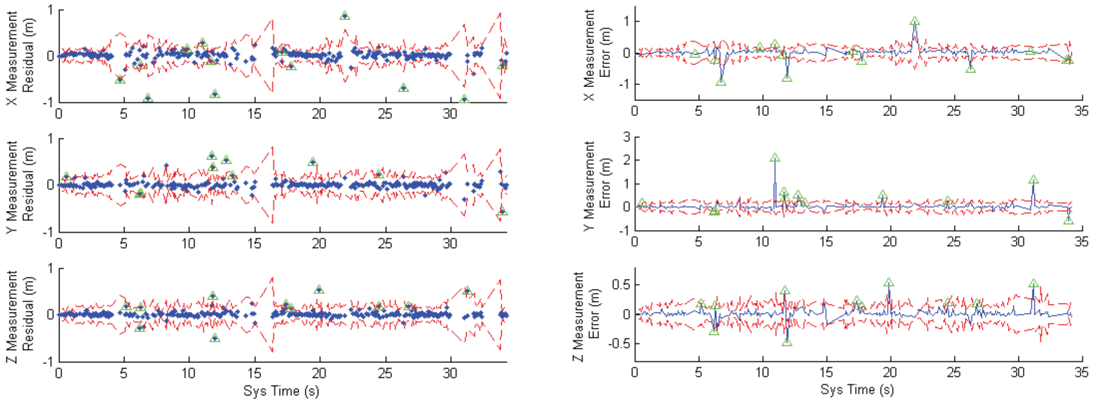
Table 4.4: Final Position Error Using Kalman Filter.

Sample	Lead RMS Error (m)	Trail RMS Error (m)
1	4.806	4.871
2	1.257	1.264
3	1.986	1.882

Firstly, it is important to inspect the behavior of the measurement residuals and ensure that measurement outliers are being rejected by the filter. Ideally, the measurement residuals should be normally distributed with zero mean and covariance determined by the Kalman filter. Therefore, an outlier can be identified by a residual value that is greater than 3-σ. Due to correlations between measurements, it was decided that if any element of the measurement vector exceeded the residual tolerance, the entire measurement for that epoch would be rejected. The measurement residuals and measurement errors are shown side-by-side in Figure 4.17. This data is from sample one, but is representative of the other trials as well.

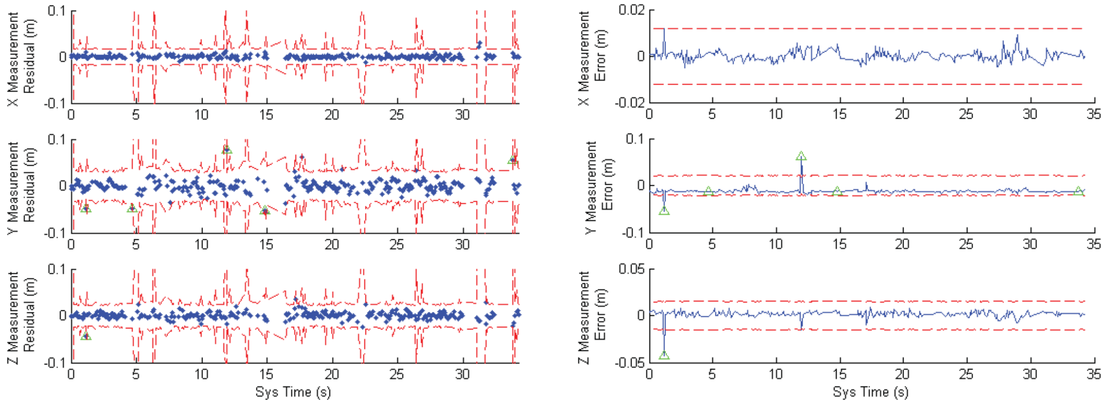


(a) Measurement Residuals $\mathbf{p}_l^w(k)$ (b) Measurement Errors $\mathbf{p}_l^w(k)$



(c) Measurement Residuals $\mathbf{p}_t^w(k)$ (d) Measurement Errors $\mathbf{p}_t^w(k)$

Figure 4.17: Kalman Filter Residuals v. Measurement Errors.



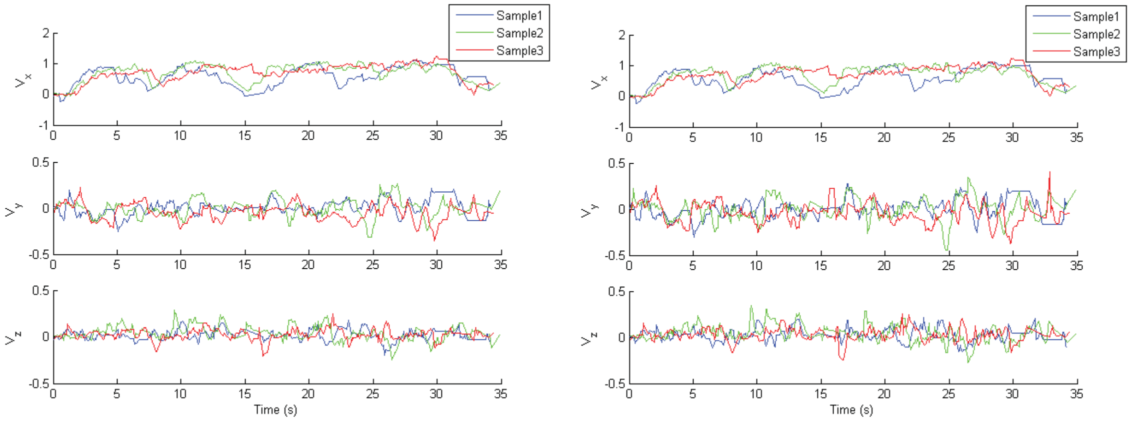
(a) Measurement Residuals $\mathbf{p}_l^w(k-1)$

(b) Measurement Errors$\mathbf{p}_l^w(k-1)$

Figure 4.18: Kalman Filter Residuals v. Measurement Errors (Continued). (Left) The residual values for each measurement (blue) with 3-σ standard deviation lines (red) as computed by the Kalman filter. (Right) The corresponding measurement (blue) with 3-σ standard deviation (red) as reported by the batch processor. Rejected measurements are shown as green triangles.

By inspection of Figures 4.17 and 4.18, it can be verified that both the measurement errors and measurement residuals are approximately zero mean and bounded by their respective standard deviation values. In addition, Figures 4.17 and 4.18 show that the true outliers are being rejected by the filter. The error in the trajectory estimate is therefore not likely related to residual monitoring. However, there also appear to be a few residuals which are rejected, but do not correspond to measurement outliers.

The rejection of feasible measurements indicates that there is a significant discrepancy between the measurement prediction based on the propagated state and the realized measurement at those epochs. This discrepancy may be related to the simplified dynamic model and the way measurements update the Kalman filter. The constant velocity model used in this thesis forces the mean velocity between epochs to remain constant and only allows the uncertainty in the velocities to grow. The value of velocity states can only be changed by an update. Due to the measurement model, the velocities are not updated by the filter directly. Instead, the velocity states are updated indirectly by the Kalman gain matrix **K** computed for each update. Figure 4.19 shows the X, Y, and Z velocities for both vehicles as computed by the Kalman filter for each sample.



(a) Lead Estimate (b) Trail Estimate

Figure 4.19: Kalman Filter Velocity Estimates. The velocities estimated by the filter are shown for the lead (a) and trail (b) vehicles

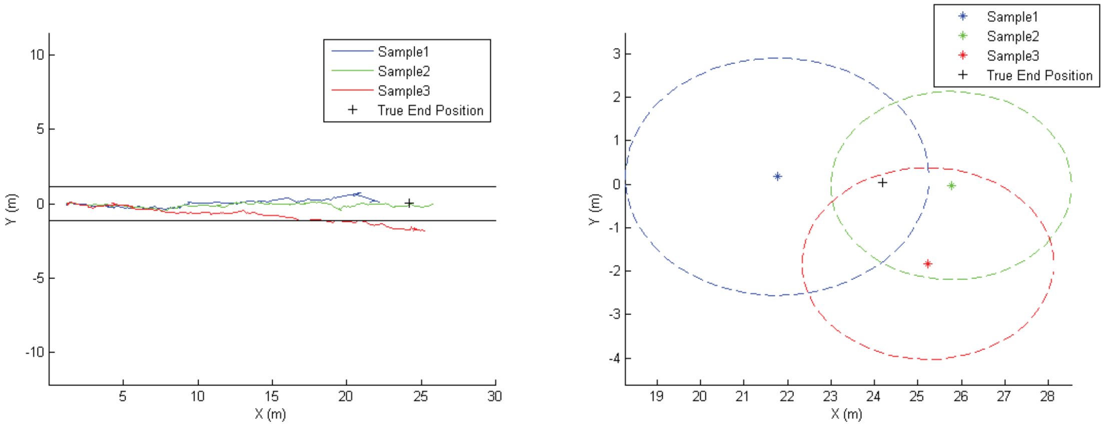
Figure 4.19 reveals the combined effects of the propagation model and measurement model. The estimated velocities in the X direction of sample one for both the lead and trail vehicles do not appear to reach a steady state as would be expected, but rather they oscillate. This oscillation is also observable in the Y and Z directions and likely accounts for the steady drift of sample two's estimated trajectory outside the walls of the test course. Inaccuracies in the estimated velocities fold into erroneous predictions by the propagation model which results in feasible measurements being rejected by residual monitoring.

4.6.6 Re-Tuning the Kalman Filter.

The results of the last section suggest that the inconsistency in the Kalman filter estimate was caused by inaccuracies in the predicted measurement as computed by the propagation model. However, Figure 4.17 did show that the measurement errors are well characterized by their computed covariance. In this section, the Kalman filter parameters will be re-tuned to reduce confidence in the propagation model and increase confidence in the measurements. To reduce the confidence in the propagation model, process noise is added to the propagation equations via the noise strength matrix for each vehicle $\mathbf{Q}_i$. The new noise strength matrix for each vehicle is

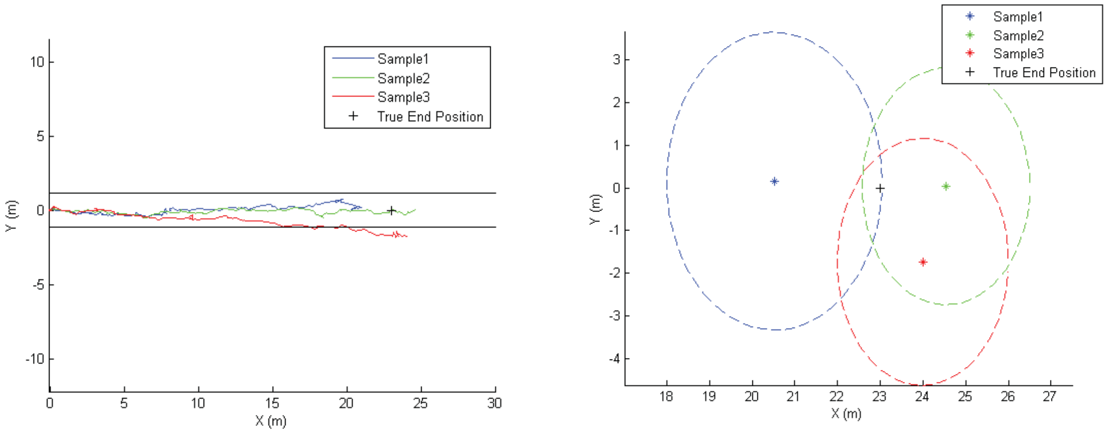
$$\mathbf{Q}_i = \begin{bmatrix} 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0.1 \end{bmatrix} \tag{4.4}$$

Increasing the propagation noise also inflates the residual covariance, which creates the potential for accepting outliers. Therefore, the residual threshold is reduced to 2.5-σ. Once again, the measurements and body angles were processed by the Kalman filter to produce estimates of the vehicle trajectories for each sample. The resulting trajectories and final positions are shown in Figure 4.20 and summarized in Table 4.5.



(a) Lead Vehicle Trajectory

(b) Final Position Estimate (Lead)



(c) Trail Vehicle Trajectory

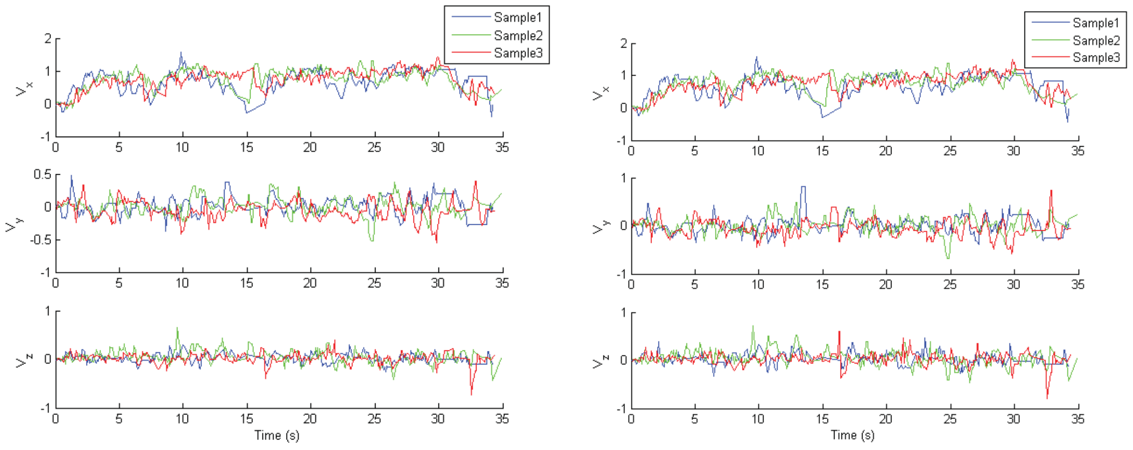
(d) Final Position Estimate (Trail)

Figure 4.20: Trajectory Estimates after Re-Tuning. The recomputed trajectories (left) and final position estimates (right) are shown for the lead and trail vehicles for each sample.

Table 4.5: Final Position Errors After Re-Tuning.

Sample	Lead RMS Error (m)	Trail RMS Error (m)
1	2.503	2.577
2	2.064	2.053
3	2.205	2.089

Although the error in the final position estimate increases for samples two and three, the error in sample one becomes consistent with the covariance calculated by the Kalman filter. This consistency does not, however, resolve the other issue pointed out in Section 4.6.5 – the velocity estimates. Figure 4.21 shows the Kalman filter estimated velocities after re-tuning.



(a) Lead Estimate

(b) Trail Estimate

Figure 4.21: Kalman Filter Velocity Estimates After Re-Tuning. The velocities estimated by the filter are shown for the lead (a) and trail (b) vehicles

While more confidence is given to the measurements, there is not a significant improvement in the velocity estimates. In fact, the velocity estimates may have become slightly worse. By increasing confidence in the measurements and increasing process noise, the filter is less able to filter out measurement noise. Instead, the filter follows the measurements as they jump around due to measurement noise.

One possible solution is to robustify the propagation model. The goal of robustifying the propagation model would be to increase the prediction accuracy of the propagation model while minimizing the required amount of process noise. Minimizing the required amount of process noise would reduce the filter's dependence on highly accurate measurements and would likely result in better velocity estimates. A more robust propagation model will be left for future work and will be briefly discussed in the next chapter.

4.6.7 Localization Improvement.

As a final comparison point, this section will show that the localization algorithm developed in this thesis has a clear performance improvement over the system designed in [18]. Recall that in [18] the on-board velocity estimation algorithm required that texture be added to the test course (in the form of tape strips on the floor) in order to return reasonable velocity estimates. Figure 4.22 compares the unfiltered trajectory estimate found by integrating the on-board velocities with the unfiltered trajectory estimate found by taking the cumulative sum of the delta-position information contained in the measurements from the batch processor on the same test course, without any modifications.

According to Figure 4.22, there is a clear improvement in the amount of motion information contained in measurements from the velocity measurements used in [18] to the feature based measurements developed in this thesis. The on-board velocities tend to underestimate the true velocity of the vehicle. This is likely related to the optical flow

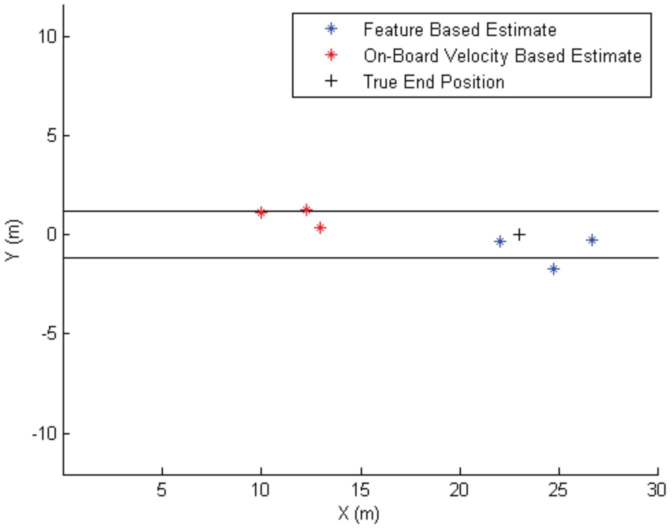


Figure 4.22: Improvements Over Previous System. The estimated final positions found by integrating on-board velocities are plotted against the estimated final positions found by taking the cumulative sum of delta-positions from the measurements developed in this thesis.

technique [8] [31] used by the on-board velocity estimation algorithm to numerically calculate the flow of an image sequence and estimate velocity. This technique depends on rich texture to detect image motion and is greatly impacted when image motion is unobservable [8] (e.g. the image sequence only captures a constant surface such as a blank wall or floor). The unmodified floor pattern in the test course often exhibits a slowly changing pattern which may cause the optical flow algorithm to determine there is no image motion and cause the performance degradation observed in Figure 4.22. The algorithms in this thesis do not suffer these same effects since they process images from the forward looking cameras which naturally contain more texture than the downward looking cameras.

4.7 Summary

In this chapter, a series of experiments were conducted to investigate the performance of the algorithms designed in the previous chapter. It was shown that the feature correspondence and triangulation algorithms are capable of determining a well localized set of matching feature points between two synchronized images and two separate time epochs. The relative camera pose estimation algorithm was shown to be capable of accurately determining the relative position and orientations of the two vehicles. Testing of the batch processor showed that the measurements it returned were well characterized by their computed covariance and are approximately Gaussian. The Kalman filtering algorithm was validated independently from the batch processor.

After linking the batch processor to the Kalman filter, it was found that the error in the estimated trajectory computed by the filter was not consistent with the true error in the trajectory estimate. This inconsistency was found to be related to the prediction accuracy of the propagation model. Inflating the process noise in the propagation model resulted in more consistent final trajectory estimates, but did not solve the issue of inaccurate velocity estimation. A more robust propagation model is needed to reckon with the inaccuracies in velocity estimation.

V. Conclusions

The goal of this research was to expand on the system developed in [18] and improve the navigation algorithm accuracy by developing an alternative method for estimating vehicle motion based on cooperative binocular stereopsis. In Chapter 3, the system developed in [18] was expanded to support multiple vehicle communications and process the synchronized image and navigation data from two ARDrone quadrotor vehicles. The complete algorithm consisted of multiple child processes including feature correspondence, feature triangulation, target tracking, and relative camera pose estimation, as well as vanishing point estimation as detailed in [18]. The triangulated feature points and relative camera positionsd were simultaneously processed by a batch algorithm to produce a single estimate of the motion of both vehicles between two time epochs. Vanishing point measurements and the batch measurement were then fused in a delayed state EKF to produce an estimate of the vehicles' trajectories.

Experiments were conducted in phases to validate each piece of the expanded navigation software. The feature matching experiment determined an appropriate tuning for the `OpenCV` feature matching objects used in this research. The relative camera pose estimation accuracy experiment characterized the errors in estimating relative camera position and orientation. The triangulation accuracy experiment characterized the errors in the feature triangulation algorithm and determined a feasible range of parallax angles to accept feature correspondences. The batch processor experiment confirmed that the true solution error was consistent with the error computed by the batch processor and that the measurements computed by the batch processor were approximately Gaussian. It was also found that enforcing a constraint on the batch solution errors improved the Kalman filter estimate. The Kalman filter experiments were able to validate the delayed state EKF algorithm for the case of a constant measurement covariance.

Integrating the batch processor and the Kalman filtering algorithm revealed a limitation of the propagation model. In order for the Kalman filter to produce a position estimate and covariance consistent with the true position error, the process noise needed to be inflated to artificially lower confidence in the propagation model. While this produced a more consistent estimate, it did not result in any significant improvement to velocity estimation which is important for autonomous navigation.

5.1 Further Testing

The experiments in Chapter 4 were able to validate the algorithms developed in this thesis for the simple case of two vehicles traveling down a single hallway in a fixed formation. This section presents a number of additional tests to further validate its operation in other scenarios and against other algorithms (e.g. SLAM).

5.1.1 Flight Path and Formation Control.

In the previous work [18], the developed algorithm was tested on a course which included a 90° turn, which verified that the vanishing point algorithm could handle a significant change in heading. The algorithms in this thesis, however, have not been tested in cases where the navigated path includes turns, changes in altitude, or when the vehicles are allowed to move in formation. These tests are needed to confirm that the algorithms designed in this thesis could be implemented in an autonomous system.

5.1.2 Runtime Concerns.

The system developed in this thesis also neglects some important runtime concerns. For instance, the system was not designed to operate during a communication loss. Precautions were taken to ensure that communications were not lost during testing. The system is also not robust to the trail vehicle losing lock on the lead vehicle and it does not possess an algorithm for formation control. In order to implement these algorithms in a real system, these capabilities must be developed and tested.

5.1.3 Feature Depletion.

The testing environment used to validate the system developed in this thesis consisted of enough features throughout the course to enable reliable feature matching. It is not reasonable to assume that the system designed in this thesis could operate effectively in an environment with sparse features or significant changes in lighting conditions. Therefore, the system designed in this thesis needs to be tested in these environments to determine its capability.

5.1.4 Comparison to State of the Art.

The cooperative binocular stereopsis algorithm developed in this thesis suggests it is an improvement over competing algorithms. In terms of its performance against SLAM algorithms, cooperative binocular stereopsis may have an advantage in computational load while maintaining an equivalent level of localization accuracy as SLAM. In terms of its performance against stereo vision algorithms, cooperative binocular stereopsis may have an advantage in localizing features in the environment since the camera baseline is allowed to be much larger than a binocular camera pair mounted on a single vehicle. In order to make a real comparison of cooperative binocular stereopsis with similar algorithms, the image and navigational data collected during experiments should be processed by these similar algorithms to determine if an advantage exists.

5.2 Future Work

There are a number of improvements that could be investigated in follow-on work. These improvements include more efficient algorithm implementations and changes to the mathematical properties of the algorithms, such as absolute positon measurements and an improved dynamic model.

5.2.1 Constrained Feature Matching.

One of the primary weaknesses in the feature matching algorithm designed in this thesis is the separation of the feature matching algorithm from the Kalman filter estimate.

The Kalman filter provides a prediction of the motion of both vehicles, along with a level of confidence in that estimate – the covariance. This motion could be used to predict a region of pixels in another image where the feature match is likely to be.

5.2.2 Absolute Position Measurements.

A limitation of the algorithms developed in this thesis, and of the various SLAM algorithms found in literature, is that the position uncertainty will eventually diverge. This is because no absolute information is given to the filter. Absolute measurements manifest in image-aided navigation as landmarks, which are known features in the environment. As Veth points out [66], the main difference between relative measurements and absolute measurements is the independence of the landmark position and and vechicle position. For absolute measurements, the landmark position and vehicle position are uncorrelated, such that the reduction in position uncertainty is greater than for relative measurements.

5.2.3 Computational Improvements.

The software developed for this thesis was not designed to optimize computational load. Computational optimization, although not specifically investigated in this thesis, is important in a real system with limited computational resources. The following subsections outline two of the most obvious areas that could use improvement.

5.2.3.1 Sparse Linear Algebra.

The Jacobians computed in the measurement processor and in the Kalman filter have an obvoius sparse structure. There are many techniques in the linear algebra literature for solving sparse systems of equations using Schur decomposition or graph based techniques. As an example, Hartley and Zisserman give an example of a partitioned Levenburg-Marquardt algorithm which takes advantage of the sparse nature of the bundle adjustment problem [28]. A similar approach could be taken within the measurement processing algorithm to partition the state of the measurement processor into vehicle positions and feature positions.

Because of the high accuracy of relative pose measurements, a simpler option would be to convert the Levenburg-Marquardt algorithm in Section 3.6 into a linearly constrained least squares problem, where the relative pose measurements form the constraints on the solution space. Since the only other measurements in the batch algorithm are feature correspondences, the problem becomes a classical linearly constrained bundle adjustment problem.

5.2.3.2 Parallelization.

The image processing functions and matrix calculations throughout the algorithms in this thesis could have been massively parallelized. A graphics processing unit (GPU) is a piece of hardware which is capable of performing many calculations in parallel. It is especially useful for image data, but can also be used to solve linear algebra equations, especially those with sparsity. OpenCV supports the creation and manipulation of matrices using a GPU and has a GPU optimized library specifically for feature matching.

5.2.4 Improved Dynamic Model.

The simplistic dynamic model used in this thesis proved to be an obstacle for accurate position estimation. A more appropriate dynamic model would include inputs in the dynamic model, leading to more accurate measurement predictions without inflating process noise. One source of dynamic model inputs are the raw inertial measurements. The measurement equations could be redefined to correspond to the bias terms of an IMU. Unfortunately, as mentioned in Chapter 3, the raw inertial measurements are not accessible. Thus, this approach would require a different platform.

Another approach would be to use the reported body angles to estimate the velocities of each vehicle. There is an approximately proportional relationship between pitch angle and forward velocity. The same relationship exists between yaw and sideways velocity. By using the corrected heading angle, the velocities of each vehicle in the navigation frame could be estimated and used to propagate the system to the next update time.

5.3 Final Conclusion

This research developed a cooperative localization algorithm based on two-view geometry which overcomes weaknesses in previous work and is able to estimate the trajectory of two independent vehicles from tracking feature points over only two time epochs. With further testing, this algorithm has the potential as an alternative to treating feature correspondences in a SLAM network. This research also provided another example of the ARDrone's utility as a platform for vision-aided navigation research.

References

[1] "Parrot AR.Drone2.0". URL http://ardrone2.parrot.com/.

[2] "ROS". URL http://www.ros.org.

[3] Adler, B. and Junhao Xiao. "Towards Autonomous Airborne Mapping of Urban Environments". *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, 77–82. 2012.

[4] Alahi, A., R. Ortiz, and P. Vandergheynst. "Freak: Fast Retina Keypoint". *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 510–517. IEEE, 2012.

[5] Bailey, T., J. Nieto, J. Guivant, M. Stevens, and E. Nebot. "Consistency of the EKF-SLAM Algorithm". *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 3562–3568. 2006.

[6] Bay, H., T. Tuytelaars, and L. Van Gool. "SURF: Speeded Up Robust Features". *Computer VisionECCV 2006*, 404–417, 2006.

[7] Bayard, D. S. and P. B. Brugarolas. "An Estimation Algorithm for Vision-Based Exploration of Small Bodies in Space". *American Control Conference, 2005. Proceedings of the 2005*, 4589–4595 vol. 7. 2005.

[8] Beauchemin, S. S. and J. L. Barron. "The Computation of Optical Flow". *ACM Computing Surveys (CSUR)*, 27(3):433–466, 1995.

[9] Bouguet, J. "Camera Calibration Toolbox for MATLAB", 2004. URL http://www.vision.caltech.edu/bouguetj/calib_doc/.

[10] Bradski, G. "The OpenCV Library". *Dr. Dobb's Journal of Software Tools*, 2000.

[11] Brown, R. G. and P. Y. C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering*. 1996.

[12] Casella, G. and R. L. Berger. *Statistical Inference*. Cengage Learning, 2 edition, 2002.

[13] Castellanos, J. A., R. Martinez-Cantin, J. D. Tards, and J. Neira. "Robocentric Map Joining: Improving the Consistency of EKF-SLAM". *Robotics and Autonomous Systems*, 55(1):21–29, 2007.

[14] Celik, K., S. J. Chung, and A. Somani. "Mono-Vision Corner SLAM for Indoor Navigation". *Electro/Information Technology, 2008. EIT 2008. IEEE International Conference on*, 343–348. 2008.

[15] Chilian, A. and H. Hirschmuller. “Stereo Camera Based Navigation of Mobile Robots on Rough Terrain”. *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 4571–4576. IEEE, 2009.

[16] Civera, J., A. J. Davison, and J. Montiel. “Inverse Depth Parametrization for Monocular SLAM”. *Robotics, IEEE Transactions on*, 24(5):932–945, 2008.

[17] Dai-xian, Z. “Binocular Vision-SLAM Using Improved SIFT Algorithm”. *Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on*, 1–4. IEEE, 2010.

[18] Dean, J. “Real-Time Heading Estimation Using Perspective Features”, 2013.

[19] Demir, G. K., R. M. Voylest, and A. C. Larson. “Motion Estimation with Cooperatively Working Multiple Robots”. *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 2, 1556–1561 vol.2. 2004.

[20] Diel, D. D., P. DeBitetto, and S. Teller. “Epipolar Constraints for Vision-Aided Inertial Navigation”. *Application of Computer Vision, 2005. WACV/MOTIONS ’05 Volume 1. Seventh IEEE Workshops on*, volume 2, 221–228. 2005.

[21] Engels, C., H. Stewnius, and D. Nistr. “Bundle Adjustment Rules”. *Photogrammetric Computer Vision*, 2, 2006.

[22] Estrada, C., J. Neira, and J. D. Tardos. “Hierarchical SLAM: Real-Time Accurate Mapping of Large Environments”. *Robotics, IEEE Transactions on*, 21(4):588–596, 2005.

[23] Fang Gao, L., Y. Gai, and S. Fu. “Simultaneous Localization and Mapping for Autonomous Mobile Robots Using Binocular Stereo Vision System”. *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, 326–330. IEEE, 2007.

[24] Fischler, M. A. and R. C. Bolles. “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. *Communications of the ACM*, 24(6):381–395, 1981.

[25] Forsyth, D. and J. Ponce. *Computer Vision: A Modern Approach*. Prentice-Hall, 2003.

[26] Gao, X., X. Hou, J. Tang, and H. Cheng. “Complete Solution Classification for the Perspective-Three-Point Problem”. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(8):930–943, 2003.

[27] Haralick, B. M., C.n Lee, K. Ottenberg, and M. Nölle. “Review and Analysis of Solutions of the Three Point Perspective Pose Estimation Psroblem”. *International Journal of Computer Vision*, 13(3):331–356, 1994.

[28] Hartley, R. I. and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, 2 edition, 2003.

[29] Hesch, J. A., D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis. "Towards Consistent Vision-Aided Inertial Navigation". *Proc. of the Int. Workshop on the Algorithmic Foundations of Robotics, Cambridge, MA, Jun*, volume 1315. 2012.

[30] Horaud, R., B. Conio, O. Leboulleux, and B. Lacolle. "An Analytic Solution for the Perspective 4-Point Problem". *Computer Vision, Graphics, and Image Processing*, 47(1):33–44, 1989.

[31] Horn, B. K. P. and B. G. Schunck. "Determining Optical Flow". *Artificial intelligence*, 17(1):185–203, 1981.

[32] Ila, V., J. M. Porta, and J. Andrade-Cetto. "Information-Based Compact Pose SLAM". *Robotics, IEEE Transactions on*, 26(1):78–93, 2010.

[33] Indelman, V. "Bundle Adjustment Without Iterative Structure Estimation and Its Application to Navigation". *Position Location and Navigation Symposium (PLANS), 2012 IEEE/ION*, 748–756. 2012.

[34] Indelman, V., P. Gurfil, E. Rivlin, and H. Rotstein. "Distributed Vision-Aided Cooperative Localization and Navigation Based on Three-View Geometry". *Aerospace Conference, 2011 IEEE*, 1–20. 2011.

[35] Indelman, V., P. Gurfil, E. Rivlin, and H. Rotstein. "Real-Time Vision-Aided Localization and Navigation Based on Three-View Geometry". *Aerospace and Electronic Systems, IEEE Transactions on*, 48(3):2239–2259, 2012.

[36] Julier, S. J. and J. K. Uhlmann. "A Counter Example to the Theory of Simultaneous Localization and Map Building". *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, 4238–4243. IEEE, 2001.

[37] Kaess, M., S. Williams, V. Indelman, R. Roberts, J. J. Leonard, and F. Dellaert. "Concurrent Filtering and Smoothing". *Information Fusion (FUSION), 2012 15th International Conference on*, 1300–1307. 2012.

[38] Konolige, K. and M. Agrawal. "FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping". *Robotics, IEEE Transactions on*, 24(5):1066–1077, 2008.

[39] Lepetit, V., F. Moreno-Noguer, and P. Fua. "Epnp: An Accurate $O(n)$ Solution to the PnP Problem". *International Journal of Computer Vision*, 81(2):155–166, 2009.

[40] Leutenegger, M. Chli, S. and R.Y. Siegwart. "BRISK: Binary Robust Invariant Scalable Keypoints". *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2548–2555. IEEE, 2011.

[41] Lim, S., T. Lee, S. Lee, S. An, and S. Oh. “Adaptive Sliding Window for Hierarchical Pose-Graph-Based SLAM”. *Control, Automation and Systems (ICCAS), 2012 12th International Conference on*, 2153–2158. 2012.

[42] Liu, Y. and Q. Dai. “Vision Aided Unmanned Aerial Vehicle Autonomy: An Overview”. *Image and Signal Processing (CISP), 2010 3rd International Congress on*, volume 1, 417–421. 2010.

[43] Lowe, D. G. “Distinctive Image Features from Scale-Invariant Keypoints”. 2004.

[44] Maybeck, P. S. *Stochastic Models, Estimation, and Control*, volume 2. Navtech Book and Software Store, 1994.

[45] Melnyk, I. V., J. A. Hesch, and S. I. Roumeliotis. “Cooperative Vision-Aided Inertial Navigation Using Overlapping Views”. *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 936–943. 2012.

[46] Monajjemi, M. “ardrone_autonomy : A ROS Driver for ARDrone 1.0 and 2.0”.

[47] Mourikis, A. I. and S. I. Roumeliotis. “Performance Analysis of Multirobot Cooperative Localization”. *Robotics, IEEE Transactions on*, 22(4):666–681, 2006.

[48] Mourikis, A. I. and S. I. Roumeliotis. “A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation”. *Robotics and Automation, 2007 IEEE International Conference on*, 3565–3572. 2007.

[49] Mourikis, A. I. and S. I. Roumeliotis. “A Dual-Layer Estimator Architecture for Long-Term Localization”. *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW ’08. IEEE Computer Society Conference on*, 1–8. 2008.

[50] Paz, L. M., J. D. Tardos, and J. Neira. “Divide and Conquer: EKF SLAM in $O(n)$”. *Robotics, IEEE Transactions on*, 24(5):1107–1120, 2008.

[51] Pinies, P. and J. D. Tardos. “Large-Scale SLAM Building Conditionally Independent Local Maps: Application to Monocular Vision”. *Robotics, IEEE Transactions on*, 24(5):1094–1106, 2008.

[52] Quigley, M., K. Conley, B.P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. “ROS: An Open-Source Robot Operating System”. *ICRA Workshop on Open Source Software*. 2009.

[53] Rekleitis, I. M., G. Dudek, and E. E. Milios. “Multi-Robot Cooperative Localization: A Study of Trade-Offs Between Efficiency and Accuracy”. *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, 2690–2695 vol.3. 2002.

[54] Rous, M., H. Lupschen, and K. F Kraiss. “Vision-Based Indoor Scene Analysis for Natural Landmark Detection”. *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 4642–4647. 2005.

[55] Rublee, E., V. Rabaud, K. Konolige, and G. Bradski. “ORB: An Efficient Alternative to SIFT or SURF”. *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2564–2571. IEEE, 2011.

[56] Sazdovski, V. and P. M. G. Silson. “Inertial Navigation Aided by Vision-Based Simultaneous Localization and Mapping”. *Sensors Journal, IEEE*, 11(8):1646–1656, 2011.

[57] Schleicher, D., L. M. Bergasa, R. Barea, E. Lopez, and M. Ocana. “Real-Time Simultaneous Localization and Mapping Using a Wide-Angle Stereo Camera”. *Distributed Intelligent Systems: Collective Intelligence and Its Applications, 2006. DIS 2006. IEEE Workshop on*, 55–60. IEEE, 2006.

[58] Shapiro, L. G. and G. C. Stockman. *Computer Vision*. Prentice-Hall, 2001.

[59] Smith, R., M. Self, and P. Cheeseman. “Estimating Uncertain Spatial Relationships In Robotics”. *Autonomous robot vehicles*, 1:167–193, 1990.

[60] Strang, G. *Introduction to Linear Algebra*. Cengage Learning, 4 edition, 2006.

[61] Suh, J., S. You, and S. Oh. “A Cooperative Localization Algorithm for Mobile Sensor Networks”. *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*, 1126–1131. 2012.

[62] Teller, S. and M. E. Antone. “Robust Camera Pose Recovery Using Stochastic Geometry”. *Proceedings of the AOIS-2001*. Citeseer, 2001.

[63] Titterton, D. H. and J. L. Weston. *Strapdown Interial Navigation Technology*, volume 207. AIAA and IEEE, 2 edition, 2004.

[64] Trajković, M. and M. Hedley. “FAST Corner Detection”. *Image and vision computing*, 16(2):75–87, 1998.

[65] Triggs, B., P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. “Bundle Adjustment: A Modern Synthesis”.

[66] Veth, M. J. *Fusion of Imaging and Inertial Sensors for Navigation*, 2006.

[67] Xu, Y., Y. Zhao, F. Wu, and K. Yang. “Error Analysis of Calibration Parameters Estimation for Binocular Stereo Vision System”. *Imaging Systems and Techniques (IST), 2013 IEEE International Conference on*, 317–320. Oct 2013.

[68] Yue, D., X. Huang, and H. Tan. “INS/VNS Fusion Based on Unscented Particle Filter”. *Wavelet Analysis and Pattern Recognition, 2007. ICWAPR ’07. International Conference on*, volume 1, 151–156. 2007.

[69] Zhang, Z. “A Flexible New Technique for Camera Calibration”. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD–MM–YYYY)* 27–03–2014	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Oct 2013–Mar 2014

4. TITLE AND SUBTITLE

Image-Aided Navigation
Using Cooperative Binocular Stereopsis

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Soeder, Justin T., Second Lieutenant, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB, OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-14-M-70

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Munitions Directorate, Air Force Research Laboratory
101 West Eglin Blvd. 140
Eglin AFB, FL 32542-6810
kevin.brink@us.af.mil
850.882.4600

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RWWI

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED

13. SUPPLEMENTARY NOTES

This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

This thesis proposes a novel method for cooperatively estimating the positions of two vehicles in a global reference frame based on synchronized image and inertial information. The proposed technique – cooperative binocular stereopsis – leverages the ability of one vehicle to reliably localize itself relative to the other vehicle using image data which enables motion estimation from tracking the three dimensional positions of common features. Unlike popular simultaneous localization and mapping (SLAM) techniques, the method proposed in this work does not require that the positions of features be carried forward in memory. Instead, the optimal vehicle motion over a single time interval is estimated from the positions of common features using a modified bundle adjustment algorithm and is used as a measurement in a delayed state extended Kalman filter (EKF). The developed system achieves improved motion estimation as compared to previous work and is a potential alternative to map-based SLAM algorithms.

15. SUBJECT TERMS

cooperative, localization, delayed state kalman filter, stereopsis

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. John F. Raquet (ENG)
U	U	U	UU	128	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-3636 x4580 john.raquet@afit.edu

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18